目　录

第 1 章　模拟和数字控制

1.1　原理

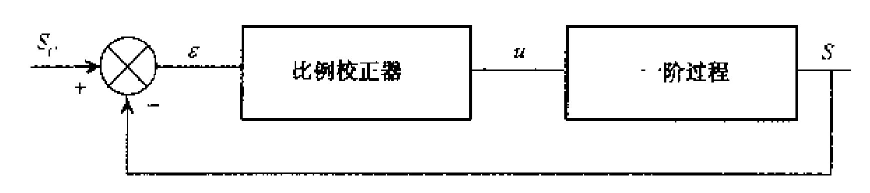


图 1-1　系统结构框图

综合应用各类校正器的目的主要有：

- 提高速度；
- 减小由干扰引起的振幅；
- 消除位置误差；
- 使原本不稳定的过程稳定。

1.2　几种主要校正器

1.2.1　比例校正器

比例校正器取决于一个简单的常量 K，表达式为：

$$C(p)=K$$

$$U(p)=K\varepsilon(p)$$

选择一个静态增益为 T_0、时间常数为 τ 的一阶过程：

$$T(p)=\frac{T_0}{1+\tau p}$$

其闭环传递函数如下所示：

$$\text{CLTF}=\frac{S(p)}{S_C(p)}=\frac{T(p)C(p)}{1+T(p)C(p)}=\frac{KT_0}{1+\tau p+KT_0}$$

或

$$\text{CLTF}=\frac{T_0'}{1+\tau' p}$$

其中

$$\begin{cases} T_0' = \dfrac{KT_0}{1+KT_0} \\ \tau' = \dfrac{\tau}{1+KT_0} \end{cases}$$

闭环系统阶次没有改变，其时间常数与 K 成反比。

如果 $KT_0 >> 1$，那么位置误差和过程控制量将趋向于零，控制量很快达到饱和点，限制了校正器单独应用的可能性。

1.2.2 积分校正器

积分校正器表达式为：

$$C(p) = \frac{1}{T_i p}$$

与比例校正器不同，这种校正器能够消除位置误差而控制量不饱和。仍以前面的一阶过程为例，闭环传递函数如下所示：

$$\text{CLTF} = \frac{T_0}{T_i p(1+\tau p)+T_0} = \frac{1}{1+\dfrac{T_i}{T_0}p+\dfrac{T_i\tau}{T_0}p^2}$$

其闭环系统的阶次比过程阶次增加了一阶，没有位置误差。该闭环系统等价于一个具有如下自然振荡频率和阻尼比的二阶系统：

$$\omega_n = \sqrt{\frac{T_0}{T_i\tau}}$$

$$\xi = \frac{1}{2}\sqrt{\frac{T_i}{T_0\tau}}$$

默认 T_i 和过程主要时间常数 τ 为同一个数量级。

1.2.3 微分校正器

微分校正器表达式为：

$$C(p) = T_d p$$

由于微分作用考虑了误差 $\varepsilon(t)$ 的变化方向和速度，故这种校正器经常和比例校正器一起使用，表达式为：

$$C(p) = K(1+T_d p)$$

仍以同样的一阶过程为例，其闭环传递函数为：

$$\text{CLTF} = \frac{T_0K(1+T_d p)}{T_0K(1+T_d p)+1+\tau p}$$

$$\text{CLTF} = \frac{T_0K}{1+T_0K} \times \frac{1+T_d p}{1+\left(\dfrac{\tau+T_0KT_d}{1+T_0K}\right)p}$$

即

$$\mathrm{CLTF}=T_0'\left(\frac{1+T_d p}{1+\tau' p}\right)$$

其中

$$\begin{cases}T_0'=\dfrac{T_0K}{1+T_0K}\\ \tau'=\dfrac{\tau+T_0KT_d}{1+T_0K}\end{cases}$$

如果 $T_0K>>1$，那么

$$\begin{cases}T_0'\approx 1\\ \tau'\approx\dfrac{\tau}{T_0K}+T_d\end{cases}$$

这类校正器可提高系统的稳定性和速度。

1.2.4 微分反馈校正器

这类校正器的缺陷是误差微分放大了干扰信号的噪声。为避免这一缺陷，可以考虑测量输出的微分形式。系统结构框图如图 1-2 所示。

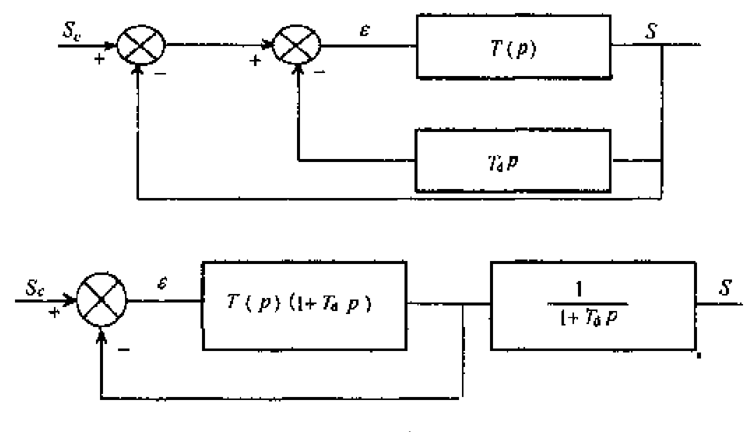


图 1-2 系统结构框图

该系统采用一个闭环的比例微分校正器和一个环外的延迟环节：

$$\frac{1}{1+T_d p}$$

这个延迟环节的作用是使暂态响应变慢。这类校正器常用在转速计反馈中。

1.2.5 相位超前校正器

相位超前校正器表达式为：

$$C(p)=\frac{1+a\tau p}{1+\tau p}$$

其中

$$a>1$$

这类校正器可以解释为一个比例微分校正器（$1+a\tau p$）乘以一个时间常数为τ的惯性环节$\frac{1}{(1+\tau p)}$。

当频率

$$\omega_{\mathrm{M}}=\frac{1}{\tau\sqrt{a}}$$

得到最大相位超前角

$$\phi_{\mathrm{M}}=\arcsin\left(\frac{a-1}{a+1}\right)$$

在高频范围内的最大振幅$C_{\mathrm{M}}=a$。

例子：a=2；τ=0.01

advance.m file

```
% Phase lead corrector
w=1:1000;
a=2;
tau=0.01;
C=(1+a*tau*j*w)./(1+tau*j*w);
disp(['Max. angle ='num2str(asin((a-1)/(a+1)))]);
disp(['frequency wm ='num2str(1/(tau*sqrt(a)))]);

Max. Angle = 0.33983
Frequency wm = 70.7107

figure(1);
semilogx(w,abs(C));
title('Phase lead corrector module');
xlabel('Frequency in rad/s');
ylabel('Module'), figure(2);
semilogx(w,angle(C));
title('Phase lead corrector argument');
xlabel('Frequency in rad/s'),ylabel('Phase in rad');

[phimax,wmax]=max(angle(C));

disp(['Max. angle ='num2str(phimax)]);
disp(['Frequency wm ='num2str(wmax)]);

Max. angel=0.33983
Frequency wm=71
```

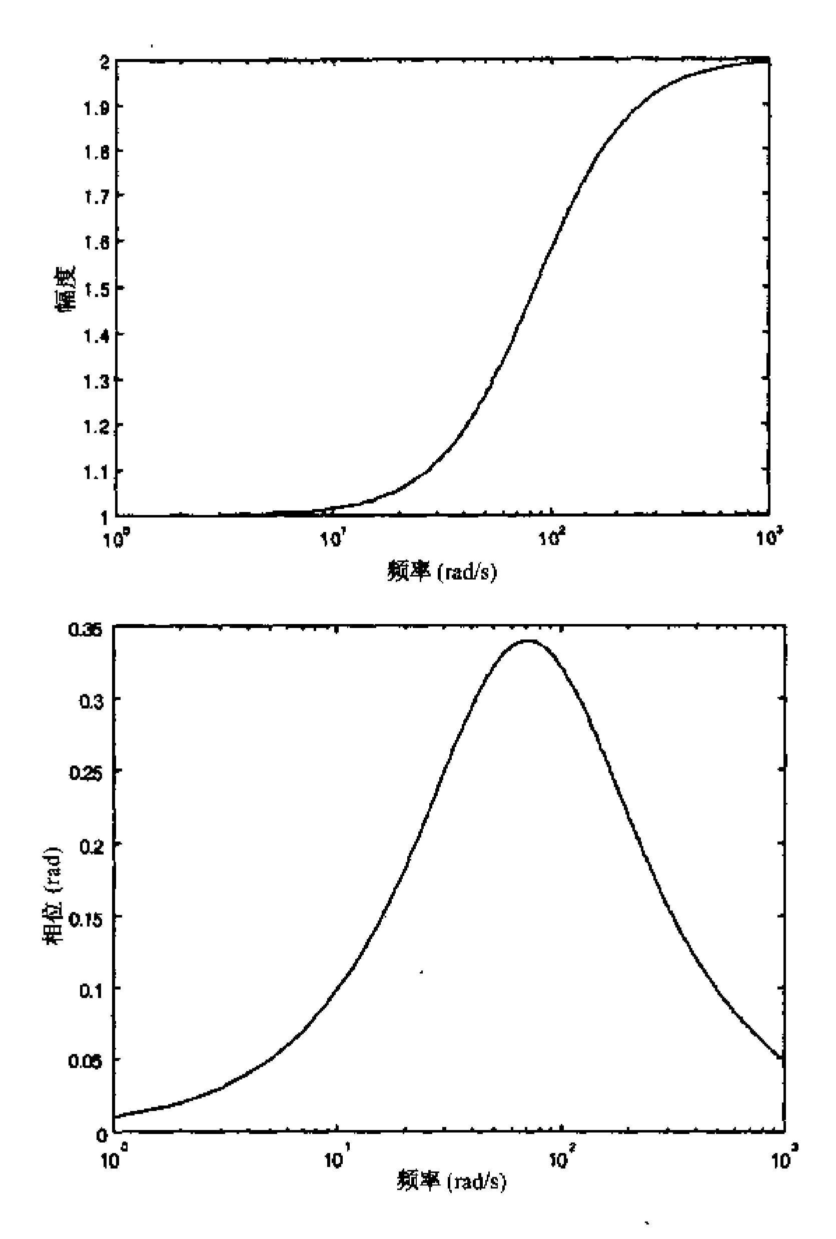


图 1-3　相位超前校正器的幅频特性和相频特性

这类校正器加宽系统的频带，因而提高了速度。把最大相位超前角调整到截止频率邻域，可以提高稳定性，产生的作用类似于微分。

1.2.6　相位滞后校正器

相位滞后校正器的表达式为：

$$C(p)=\frac{1+\tau p}{1+a\tau p}$$

其中 $a>1$。

这类校正器的波特图和相位超前校正器的波特图对称。

当频率

$$\omega_{\mathrm{M}}=\frac{1}{\tau\sqrt{a}}$$

得到最大相位滞后角

$$\phi_M = -\arcsin\left(\frac{a-1}{a+1}\right)$$

在高频范围内的最小振幅 $C_M = \frac{1}{a}$。

例子：$a=2$；$\tau = 0.01$

delay.m file

```
% Phase lag corrector
w=1:1000;
a=2;
tau=0.01;
C=(1+tau*j*w)./(1+a*tau*j*w);
disp(['Max. delay ='num2str(-asin((a-1)/(a+1)))]);
disp(['Frequency wm ='num2str(1/(tau*sqrt(a)))]);

Max. Delay =-0.33984
Frequency wm = 70.7107

figure(1);
semilogx(w,abs(C));
title('Phase lag corrector modulus');
xlabel('Frequency in rad/s');
ylabel('Modulus');
figure(2);
semilogx(w,angle(C));
title('Phase lag corrector argument');
xlabel('Frequency in rad/s');
ylabel('Phase in rad');
[phimax,wmax]=min(angle(C));
disp(['Max. delay ='num2str(phimax)]);
disp(['Frequency wm ='num2str(wmax)]);

Max. angel=-0.33983
Frequency wm=71
```

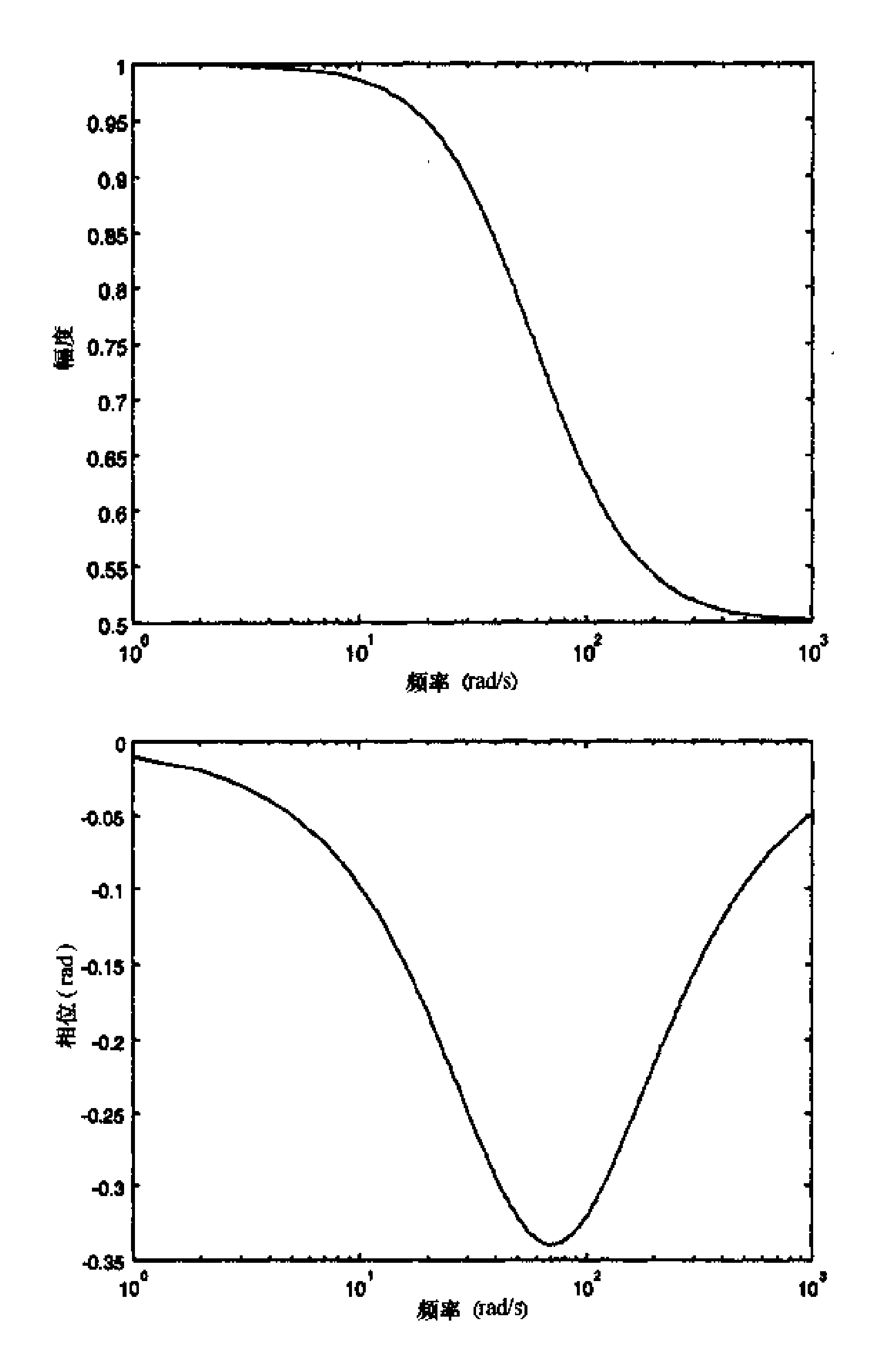


图 1-4 相位滞后校正器的幅频特性和相频特性

这类校正器的作用是减小高频振幅。在低频段要尽量消除由增加的相位延迟产生的干扰。这类校正器类似于积分校正器。

1.2.7 PID 控制器

PID 控制器是把比例、积分和微分作用结合起来，以利用其各自的优点的一类控制器，表达式为：

$$C(p)=K\left(1+T_{\mathrm{d}}p+\frac{1}{T_{i}p}\right)=\frac{K}{T_{i}p}(1+T_{i}p+T_{i}T_{\mathrm{d}}p^{2})$$

方程中含有:

- 一个零极点;
- 一阶微分作用$T_i p$;
- 二阶微分作用$T_i T_d p^2$。

记

$$\tau = \sqrt{T_i T_d}$$

得到

$$C(p) = \frac{K}{T_i p}(1 + 2\xi\tau p + \tau^2 p^2)$$

其中

$$\xi = \frac{1}{2}\sqrt{\frac{T_i}{T_d}}$$

例子: $\xi = 0.5; T_i = T_d; K = 10$

pid.m file

```
% PID corrector

w=1:10000;
Ti=0.01;
Td=0.01;
K=10;
C=K*(1+Td*j*w+1./(Ti*j*w));

figure(1);
loglog(w,abs(C));
title('PID corrector modulus');
xlabel('Frequency in rad/s');
ylabel('Modulus');

figure(2);
semilogx(w,angle(C));
title('PID corrector argument');
xlabel('Frequency in rad/s');
ylabel('Phase in rad');
```

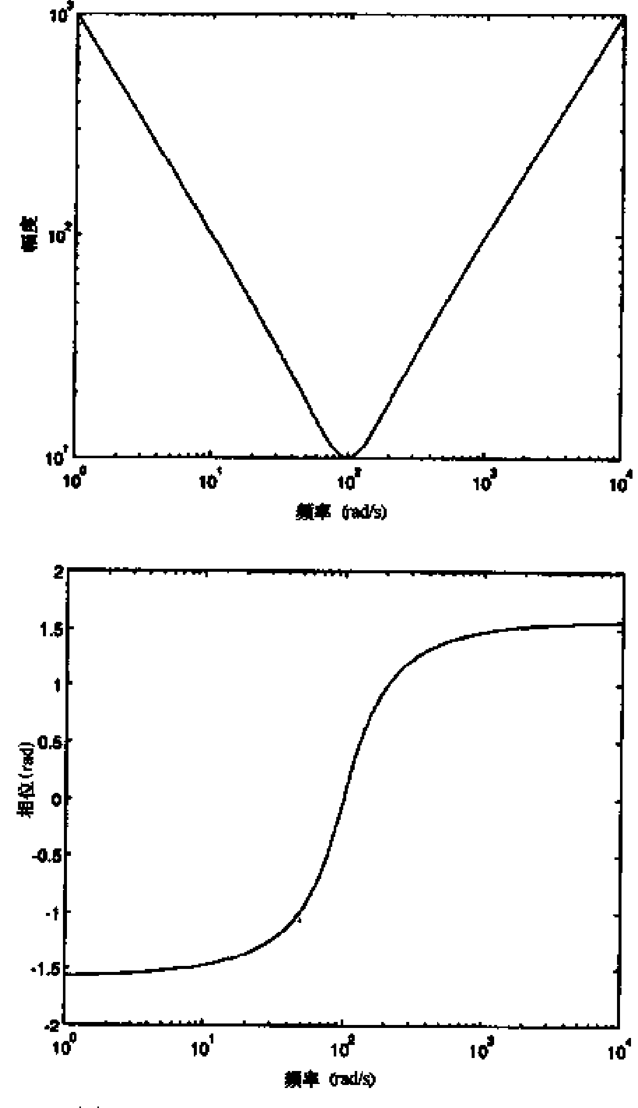


图 1-5　PID 控制器的幅频特性和相频特性

由图 1-5 可以看出，在高频段相位超前，在低频段相位滞后，所以能够增加伺服系统的增益 K 和自然振荡频率 ω_n。对于 PID 校正，首先使得时间常数 τ 满足：

$$\frac{1}{\tau} < \omega_r$$

其中 ω_r=过程共振频率。然后计算符合条件的 ξ。另一种快速的方法是取 $T_i=\tau/10$ 和 $T_d=0.25\ T_i$，其中 τ 为过程时间常数。上述 PID 具有放大高频信号（高波段噪声）的缺点，通常用得更多的是添加了滤波器的版本：

$$C(p)=K\left(1+T_d p+\frac{1}{1+T_i p}\right)\times\frac{1}{1+\dfrac{T_d}{N}p}$$

一般选择 N=10。

pid_filter.m file

```
% PID filtering corrector

w=1:10000;
```

```
Ti=0.01;
Td=0.01;
K=10;
N=10;
C=K*(1+Td*j*w+1./(Ti*j*w)).*(1./(1+Td*j*w/N));

figure(1);
loglog(w,abs(C));
title('Modulus of the PID filtered corrector');
xlabel('Frequency in rad/s');
ylabel('Modulus');

figure(2);
semilogx(w,angle(C));
title('Argument of the PID filtered corrector');
xlabel('Frequency in rad/s');
ylabel('Phase in rad');
```

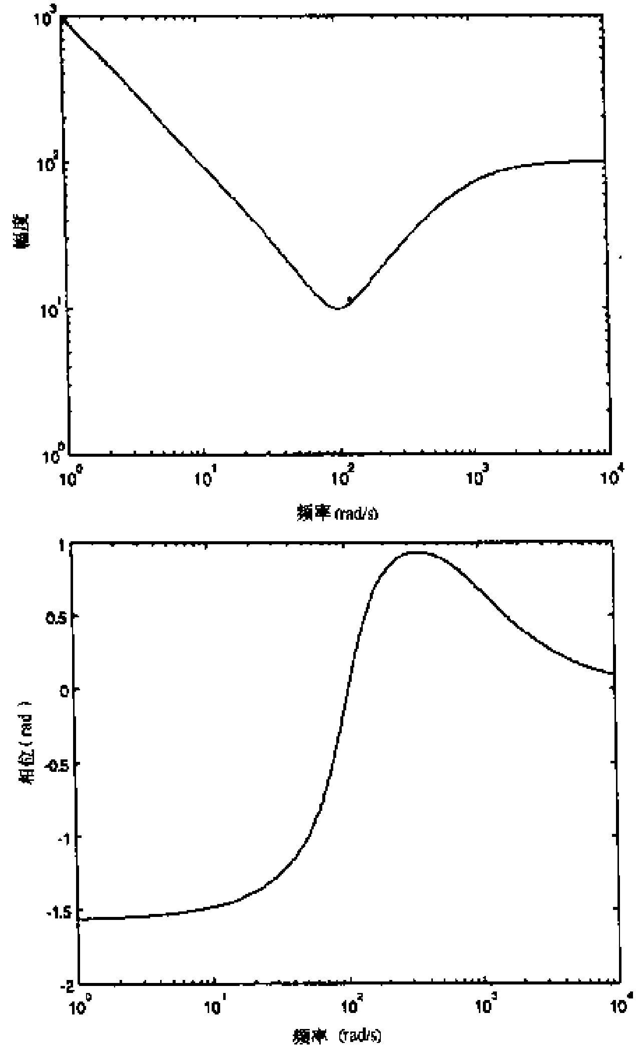


图 1-6　PID 滤波校正器的幅频特性和相频特性

1.2.8　前馈校正

前馈可直接引入一个时间函数，函数位置不同于比例微分校正器是在误差 ε 之前，而是在输入之前。

◆　前馈-随动系统（见图 1-7）

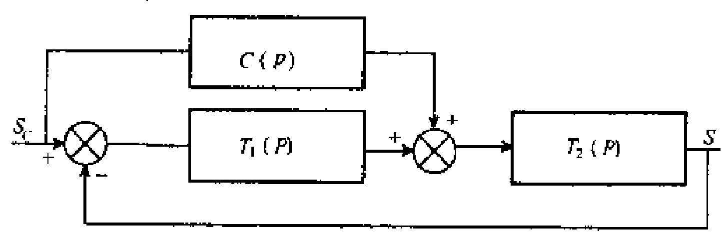


图 1-7　前馈-随动系统结构框图

$$\frac{S(p)}{S_C(p)}=\frac{T_2(p)[C(p)+T_1(p)]}{1+T_1(p)T_2(p)}$$

如果选

$$C(p)=\frac{1}{T_2(p)}$$

那么

$$\frac{S(p)}{S_C(p)}=1$$

要采用该方法，校正器通常需要分子的阶数比分母高一阶，这在实际中是不可能的。所以，只能试着采用近似的方法。

◆　前馈-校正系统（见图 1-8）

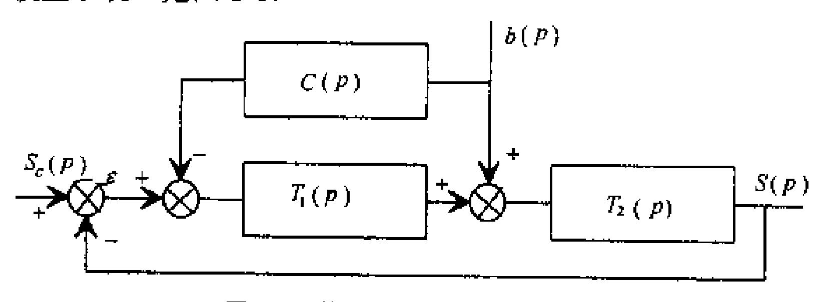


图 1-8　前馈-校正系统结构框图

这个过程需要主要干扰信号的先验知识。

当

$$C(p)T_1(p)=1$$

时可消除如图 1-8 所示的预测链的干扰影响。这个系统具有与随动系统同样的概念缺陷。

1.2.9　PIR 校正器，纯滞后系统

这类校正器应用于传递函数如下的系统：

$$T(p)=KG(p)e^{-p\tau}$$

其中 τ 为延迟秒数。如果 τ 很小，可以用一个相位超前环节来补偿。如果情况不是如此，

就可能要考虑 PIR 校正器（也称为 Smith 预测器），见图 1-9。

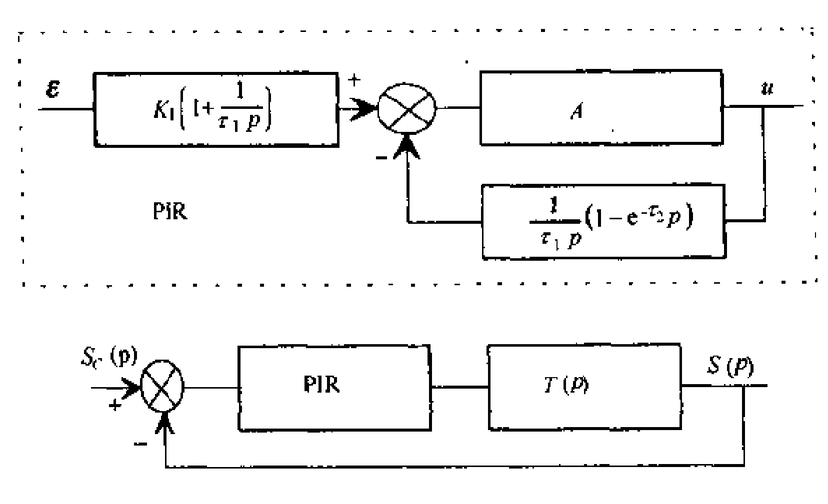


图 1-9　系统结构框图

Smith 校正器的传递函数为

$$\frac{u(p)}{\varepsilon(p)}=\frac{K_1 A(1+\tau_1 p)}{\tau_1+A(1-e^{-\tau_2 p})}$$

其闭环传递函数为

$$\frac{S(p)}{S_C(p)}=\frac{K_1 KA(1+\tau_1 p)e^{-p\tau}}{K_1 KA(1+\tau_1 p)e^{-p\tau}+(1+\tau p)[\tau_1 p+A(1-e^{-\tau_2 p})]}$$

调整参数

$$\begin{cases} K_1=\dfrac{1}{K} \\ \tau_1=\tau_2=\tau \end{cases}$$

得到

$$\frac{S(p)}{S_C(p)}=\frac{e^{-p\tau}}{1+\dfrac{\tau}{A}p}$$

增益 A 可以校正闭环系统的响应时间。

1.3　模拟校正器离散化

校正器 C（p）的离散需要对误差 ε 和通常阶数为 0 的输出模块进行采样。从模拟过程到采样系统的变换过程主要是把 p 转换为 z。这个过程有几种近似方法，最好的是梯形近似法叫做 Tustin 或双线性变换。变量 p 和变量 z 的关系是 $z=e^{pT_s}$，其中 T_s 为采样周期。用下式代替 p 可以得到 $C(z)$的表达式：

$$p=\frac{1}{T_s}\ln(z)$$

由于得到的结果难以应用，所以只采用 z 展开后的第一项：

$$\ln(z)=2\left(\omega+\frac{\omega^3}{3}+\cdots\right)$$

其中

$$|\omega|=\left|\frac{1-z^{-1}}{1+z^{-1}}\right|<1$$

即如果考虑等式的第一阶

$$p=\frac{1}{T_s}\ln(z)=\frac{2}{T_s}\times\frac{1-z^{-1}}{1+z^{-1}}$$

由上述表达式得：

$$z=\frac{2+pT_s}{2-pT_s},|z|=1\forall\omega$$

由此变换后虚轴 $p=\mathrm{j}\omega$ 变换为单位圆周。

双线性变换具有保持系统稳定性的优点，但缺点是 z 平面内的频率和 p 平面内的频率没有线性关系。实际上，

$$p=\mathrm{j}\omega=\frac{2}{T_s}\times\frac{z-1}{z+1}$$

即

$$p=\frac{2}{T_s}\times\frac{\mathrm{e}^{\mathrm{j}\omega' T_s}-1}{\mathrm{e}^{\mathrm{j}\omega' T_s}+1}=\frac{2}{T_s}\times\frac{\left(\mathrm{e}^{\frac{\mathrm{j}\omega' T_s}{2}}-\mathrm{e}^{\frac{\mathrm{j}\omega' T_s}{2}}\right)}{\left(\mathrm{e}^{\frac{\mathrm{j}\omega' T_s}{2}}+\mathrm{e}^{\frac{\mathrm{j}\omega' T_s}{2}}\right)}$$

简化得：

$$p=\frac{2\mathrm{j}}{T_s}\tan(\frac{\omega' T_s}{2})$$

其中 $p=\mathrm{j}\omega$，则表达式为：

$$\omega=\frac{2}{T_s}\tan(\frac{\omega' T_s}{2})$$

如果 $\omega' T_s<<1$，那么：

$$\tan(x)\approx x,\omega=\omega'$$

在低频段频率是线性关系。否则，要使 ω 以 1 个特殊值 ω_0 对应于 ω'，必须考虑比例系数 k：

$$k=\frac{\omega_0}{\frac{2}{T_s}\tan\left(\frac{\omega' T_s}{2}\right)}$$

1.4 校正系统的稳定性

1.4.1 一般稳定性条件

设 $T(p)$为过程传递函数，表达式为

$$T(p)=\frac{N(p)}{D(p)}$$

其中

$$D(p)=K(p-p_1)(p-p_2)\dots(p-p_n)$$

它可分解为

$$T(p)=\left[\frac{A_1}{p-p_1}+\frac{A_2}{p-p_2}+\cdots+\frac{A_n}{p-p_n}\right]N(p)$$

对每一个实极点 p_k，其响应为：

$$A_k e^{p_k t}$$

对每一个复极点 $p_k=a\pm jb$，其响应为：

$$A_k e^{at}\cos(\omega t+\phi)$$

只有当每个指数响应都收敛到 0 时，系统稳定。所以，当每个极点的实数部分是严格负数时，系统稳定。

1.4.2 奈奎斯特准则

首先画出开环传递函数的奈奎斯特图（见图 1-10）。让 p 在-∞到+∞之间变化并从右边穿越零点，如果开环传递函数的奈奎斯特图不包含-1 点，那么闭环系统稳定。

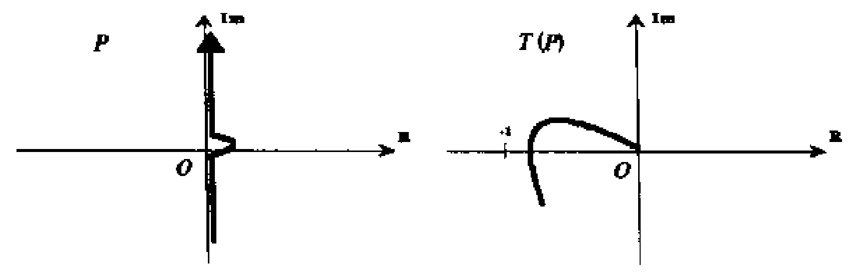


图 1-10 开环传递函数的奈奎斯特图

注意：对于 $T(p)=\frac{T_0}{p^k(\dots)}$，$p=0$ 时，曲线趋于无限，所以应让 p 从 0+顺时针旋转 $k\pi$变为 0-。

1.4.3 离散系统稳定性

考虑一个离散系统传递函数为：

$$T(z)=\frac{k}{(z-z_0)(z-z_1)}$$

系统的单位阶跃响应为：

$$y(z)=\frac{z}{z-1}\times\frac{k}{(z-z_0)(z-z_1)}$$

两边同除以 z 并写成部分分式形式：

$$\frac{y(z)}{z}=\frac{a}{z-1}+\frac{b}{z-z_0}+\frac{c}{z-z_1}$$

如果回到原时域方程：

$$y(k)=au(k)+bz_0^k+cz_1^k$$

易得如果响应收敛，则系统稳定。所以每一项都收敛时，得到：

$$|z_n|<1$$

1.5 例子

1.5.1 应用 MATLAB 函数

这里应用 MATLAB“控制系统工具箱”的一些函数。

◆ 传递函数的编写和显示

example1_chap1.m file

```
% Transfert function definition
num=[10 10];
den=[1 2 1 ];
sys1=TF(num,den);
printsys(num,den,'p');

num/den=
10 p + 10
   ------------------
   p^2 + 2 p + 1
```

◆ 零点、极点和增益计算

```
% Return zeros, poles and gain in vectors
[z,p,k]=zpkdata(sys1,'v')

z = -1

p= -1
   -1

k=10
```

◆ 传递函数的零、极点表示

```
% Representation in zeros-poles-gain form
Zsys1=zpk(sys1)

Zero/pole/gain:
10(s+1)
-----------
(s+1)^2
```

◆ 传递函数波特图

```
% Bode diagrams of the transfer function
figure(1);
bode(sys1,{0.1,50});
Title('Bode diagram')
xlabel('Frequency in rad/s')
ylabel('Phase in degrees ; Modulus in dB');
```

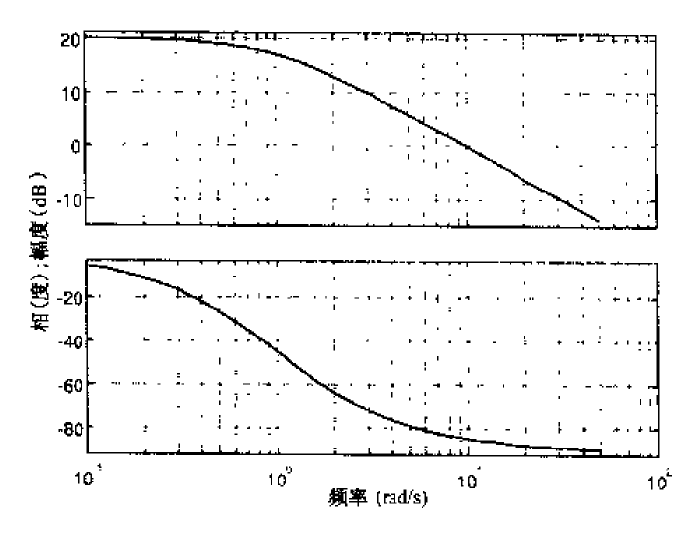


图 1-11　传递函数波特图

◆　传递函数阶跃响应

```
figure(11);
[y,t]=step(sys1);
line(t,y),grid;
Title('Response to a 1V step')
xlabel('Time')
ylabel('Amplitude in V ');
```

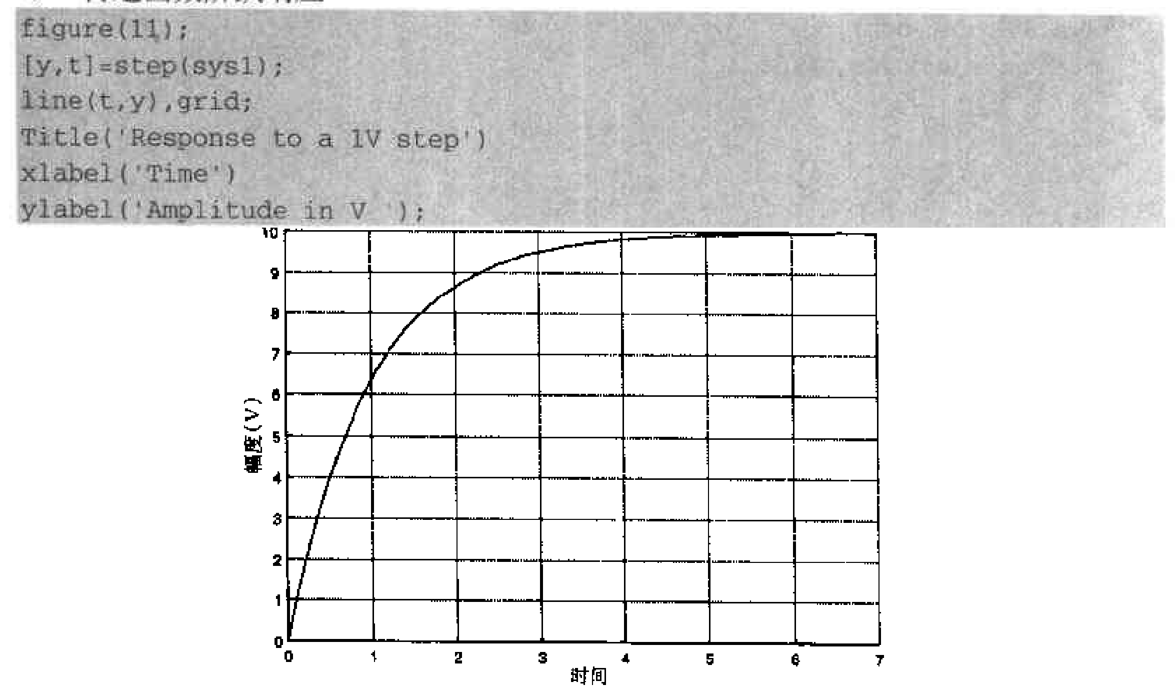


图 1-12　传递函数阶跃响应曲线

◆　传递函数脉冲响应

```
% Transfert function impulse response
figure(12);
[y,t]=impulse(sys1);
line(t,y),grid;
Title('Response to a 1V impulse')
xlabel('Time')
ylabel('Amplitude in V');
```

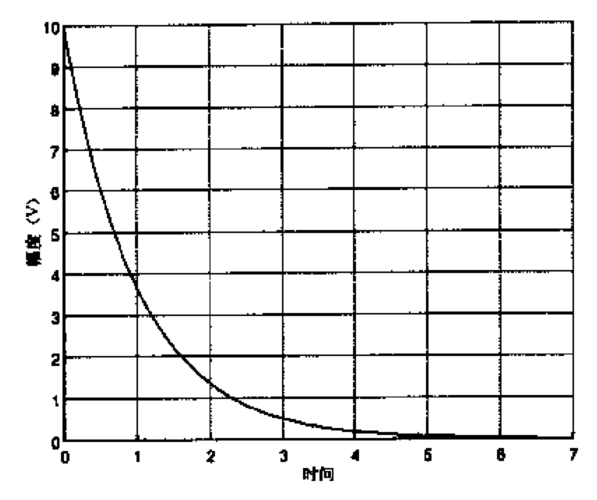


图 1-13　传递函数脉冲响应曲线

◆　Tustin 法离散化

采样时间 T_s=0.1s 时连续系统的离散化。

```
Te=0.01;
sys1d=c2d(sys1,Te,'tustin')

Transfer function:
0.04975 z^2 + 0.000495z - 0.04926
---------------------------------------------
z^2 - 1.98 z + 0.9802

Sampling time: 0.01
```

◆　离散系统脉冲响应

```
b=[0.04975 0.000495 -0.04926];
a=[1 -1.98 0.9802];
fe=1/Te;
N=400;
[h,t]=impz(b,a,N,fe);
figure(13);
plot(t,h),grid;
title('Discrete system impulse response');
xlabel('Time');
ylabel('Amplitude in V');
```

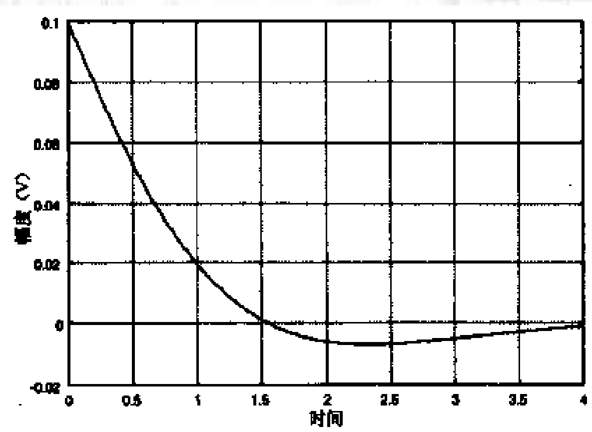


图 1-14　离散系统脉冲响应曲线

◆ 离散系统的奈奎斯特图

```
figure(2);
nyquist(sys1d),grid;
```

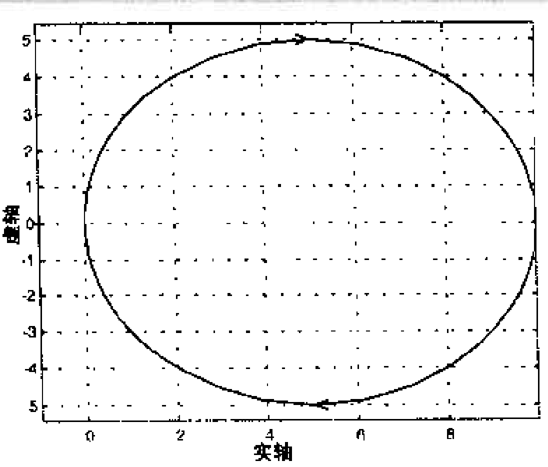


图 1-15 离散系统的奈奎斯特图

◆ **连续系统与离散系统的相位和增益的范围**

```
[mg,mp,wg,wp]=margin(sys1)

mg = Inf
mp = 95.7392
wg = NaN
wp = 9.9499

[mgd,mpd,wgd,wpd]=margin(sys1d)

mgd = Inf
mpd = 95.7392
wgd = NaN
wpd = 9.9417
```

设控制对象为模拟过程传递函数 sys1。

```
% sys1 discrete process with unit feedback.
[num2,den2]=feedback(num,den,1,1,-1);
sys2=TF(num2,den2);
printsys(num2,den2,'p');

num/den=
10 p + 10
------------------
p^2 + 12p + 11

% Static gain calculation = k/k+1 i.e. 10/11

A0=dcgain(num2,den2);
disp(['Static gain ='num2str(A0)]);

Static  gain = 0.90909
```

◆ 该闭环系统的 1V 单位阶跃响应

```
figure(3);
[y,t]=step(sys2);
line(t,y),grid;
Title('Response to a 1V step')
xlabel('Time')
ylabel('Amplitude in V ');
```

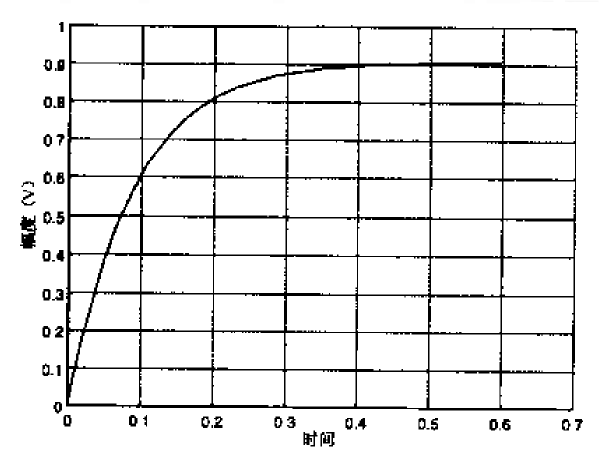


图 1-16　闭环系统的 1V 单位阶跃响应曲线

图 1-16 的几点说明：

- 响应时间是 0.1s，稳态增益是 10/11；
- 用一个 PID 校正器校正过程 sys1；
- 积分环节选择 T_i=T/10=0.1s，其中 T=过程的主要时间常数=1s；
- 微分环节选择 T_d=T_i/4=0.025 s；
- 比例环节 K=10，也可以根据伺服系统的动态特性进行修正。

```
% sys1 process with PID corrector
% PID corrector
Ti=0.1;
Td=0.025;
K=10;
num3=[K*Ti*Td K*Ti K];
den3=[Ti 0];
sys3=tf(num3,den3);

% Direct chain = PID corrector + process
sys4=sys1*sys3;

% Closed-loop system
sys5=feedback(sys4,1,1,-1);

% Indicial response
figure(4);
[y,t]=step(sys5);
line(t,y),grid;
Title('Response to a 1V step')
```

```
xlabel('Time')
ylabel('Amplitude in V ');
```

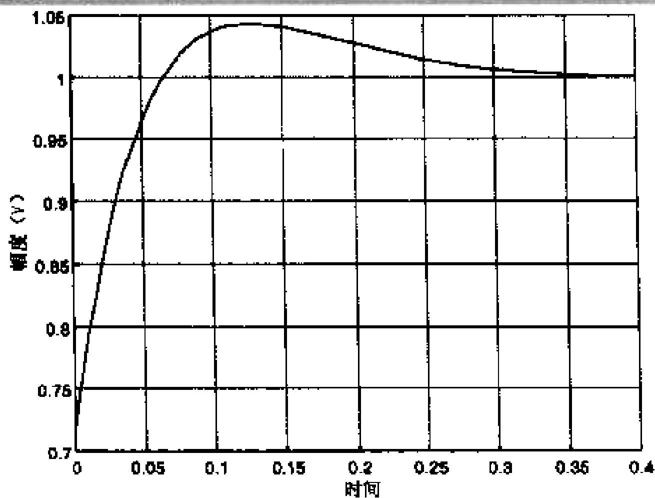


图 1-17　闭环系统的 1 V 单位阶跃响应曲线

图 1-17 的几点说明：

- 响应时间为 0.05 s；
- 是未校正过程的 5 %；
- 微分环节使响应过程稳定，而积分环节消除了稳态误差。

1.5.2　应用 PIR 校正器

以一个具有 Strejc 法确定的纯延迟环节的过程为例。纯延迟发生在具有运转间隙机械部分（减速器）的过程中。

过程的辨识模型如下：

$$T(p)=\frac{K}{1+2Tp}\mathrm{e}^{-Tp}$$

文件 Example2_chap1.m 计算这个过程相关的 PIR 校正器。PIR 校正器需要 3 个传递函数。

- 第一个传递函数：

$$G_1(p)=K_1\left(1+\frac{1}{T_1 p}\right)$$

其中 $T_1=T$ 且 $K_1=\dfrac{1}{K}$。

- 第二个传递函数：

$$G_2(p)=\frac{1-\mathrm{e}^{-T_2 p}}{T_1 p}$$

其中 $T_2=T$。

- 第三个传递函数：

$$G_3(p)=A$$

用 Pade 法近似代替传递函数中的纯延迟环节。

example2_chap1.m file

```
% PIR analog corrector
K=2.5;
T=0.1;

% Pure delay Pade approximation
[num,den]=pade(T,20);  % Approximation order = 20
sys=tf(num,den);
num1=K;
den1=[2*T 1];
sys1=tf(num1,den1);
Proc=series(sys,sys1);

% Process indicial response
figure(1);
t=(0:0.01:1.2);
[y,x]=step(Proc,t);
line(x,y),grid;
Title('Process indicial response with pure delay');
xlabel('Time');
ylabel('Amplitude in V');
```

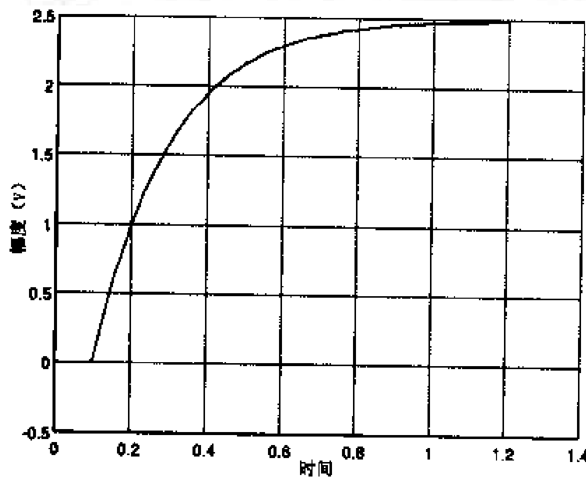


图 1-18　具有纯延迟环节过程的指数响应曲线

```
% 5% response time
vmax=max(y);
i=find((y-0.95*vmax)>0);
disp(['Response time = 'num2str(x(i(1,1)))]);

% PIR corrector calculation
K1=1/K;
T1=T;
T2=T;
A=50;
num3=[K1*T1 K1];
den3=[T1 0];
G1=tf(num3,den3);
```

```
sys4=tf(1,den3);
G2=sys4*(1-sys);
sys6=feedback(A,G2,-1);
PIR=sys6*G1;

% Closed-loop transfert function calculation
FTBO=PIR*Proc;
FTBF=feedback(FTBO,1,-1);

% Plot of the corrected process indicial response
figure(2);
[y,x]=step(FTBF,t);
line(x,y),grid;
title('Indicial response of the closed-loop transfer function');
xlabel('Time');
ylabel('Amplitude in V');
```

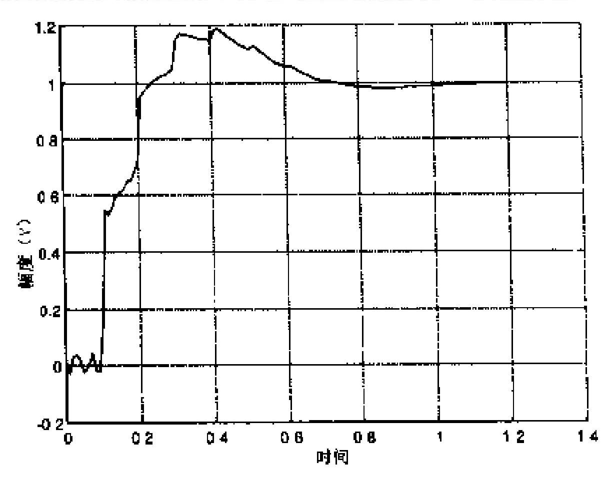


图 1-19　闭环系统的指数响应曲线

校正过程指数响应的畸变是由于纯延迟的近似替换造成的（见图 1-19）。

1.6　LQ，LQI，线性二次项控制

下面提到的 LQI 控制是基于二次项性能指标的最小化，其中二次项是由控制信号 $u(t)$ 的变化乘以一定的系数后与预测误差 $e(t+1)$相加组成的。该过程不包含积分环节，通过包含控制量变化 $\Delta u(t) = u(t)-u(t-1)$来达到和积分环节同样的效果。利用预测误差 $e(t+1)$可以得出一步预测控制律。这一节主要讲述这一控制律在单变量和多变量过程中的应用，对每一过程又分为有积分环节和无积分环节两种过程模型。

1.6.1　单变量过程的 LQI 控制

1.6.1.1　无积分器模型

最小化性能指标 J 等于 t 采样时刻控制量变化平方加上$(t+1)$时刻预测误差平方，表达式为：

$$J = e(t+1)^2 + R\Delta u(t)^2$$

性能指标最小化即求满足下式的最优控制量变化：

$$\frac{\partial J}{\partial(\Delta u)}=0$$

用控制增量Δ$u(t)$表示预测误差 $e(t+1)$能够推导出一步采样预测控制并且能包含积分环节，过程模型如下(通常以 ARMA 形式表示)：

$$H(z)=\frac{\boldsymbol{B}(z)}{\boldsymbol{A}(z)}=z^{-1}\frac{\boldsymbol{B}'(z)}{\boldsymbol{A}(z)}=z^{-1}\frac{\sum_{k=1}^{m}b_k z^{-k+1}}{1-\sum_{i=1}^{n}a_i z^{-i}}$$

这个模型的预测输出方程为：

$$\hat{y}(t+1)=\boldsymbol{A}^*(z)y(t)+\boldsymbol{B}^*(z)u(t-1)+b_1u(t)$$

其中

$$\boldsymbol{A}^*(z)=z[1-\boldsymbol{A}(z)],\boldsymbol{B}^*(z)=z[\boldsymbol{B}'(z)-b_1]$$

如果 $r(t)$为跟踪信号，则 J 性能指标可表示为：

$$J=[r(t+1)-\hat{y}(t+1)]^2+R\Delta u(t)^2$$

把预测输出方程写成包含控制变量Δ$u(t)$的形式：

$$\hat{y}(t+1)=[1+\boldsymbol{A}^*(z)]y(t)-\boldsymbol{A}^*(z)y(t-1)+\boldsymbol{B}^*(z)u(t-1)+b_1\Delta u(t)$$

则 J 性能指标为：

$$J=\left\{r(t+1)-[1+\boldsymbol{A}^*(z)]y(t)+\boldsymbol{A}^*(z)y(t-1)-\boldsymbol{B}^*(z)\Delta u(t-1)-b_1\Delta u(t)\right\}$$

等式两边同时对Δ$u(t)$求导，得出最优控制变量：

$$\Delta u(t)=\frac{b_1}{b_1^2+R}\left\{r(t+1)-[1+\boldsymbol{A}^*(z)]y(t)+\boldsymbol{A}^*(z)y(t-1)-\boldsymbol{B}^*(z)\Delta u(t-1)\right\}$$

则 t 时刻的控制量 $u(t)=u(t-1)+\Delta u(t)$。

1.6.1.2　有积分器模型

这种情况下，可通过性能指标最小化直接求出过程控制量 $u(t)$。预测输出方程如下：

$$\hat{y}(t+1)=\boldsymbol{A}^*(z)y(t)+\boldsymbol{B}^*(z)u(t-1)+b_1u(t)$$

最小化性能指标等式

$$J=\left\{r(t+1)-\boldsymbol{A}^*(z)-\boldsymbol{B}^*(z)\Delta u(t-1)-b_1u(t)\right\}^2+Ru(t)^2$$

等式两边同时对 $u(t)$求导，得到过程的最优控制量

$$u(t)=\frac{b_1}{b_1^2+R}\left\{r(t+1)-\boldsymbol{A}^*(z)y(t)-\boldsymbol{B}^*(z)u(t-1)\right\}$$

1.6.2　多变量过程的 LQI 控制

为了使推导过程简捷明了，以两个输入、两个输出的多变量过程为例（见图 1-20)。



图 1-20　多变量过程

设输入输出关系由如下 ARMA 形式的传递函数表示：

$$H_{jk}(z)=\frac{\boldsymbol{B}_{jk}(z)}{\boldsymbol{A}_j(z)}=z^{-1}\frac{\boldsymbol{B}'_{jk}(z)}{\boldsymbol{A}_j(z)}$$

所以

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}=\begin{bmatrix} H_{11}(z) & H_{12}(z) \\ H_{21}(z) & H_{22}(z) \end{bmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

类似于单变量过程，该过程的预测输出方程如下：

$$\begin{pmatrix} y_1(t+1) \\ y_2(t+1) \end{pmatrix}=\begin{bmatrix} \boldsymbol{A}_1^*(z) & 0 \\ 0 & \boldsymbol{A}_2^* \end{bmatrix}\begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix}+\begin{bmatrix} \boldsymbol{B}^*{}_{11}(z) & \boldsymbol{B}^*{}_{12}(z) \\ \boldsymbol{B}^*{}_{21}(z) & \boldsymbol{B}^*{}_{22}(z) \end{bmatrix}\begin{pmatrix} u_1(t-1) \\ u_2(t-1) \end{pmatrix}+\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}\begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix}$$

也可写成下列形式：

$$y(t+1)=\boldsymbol{A}^*(z)y(t)+\boldsymbol{B}^*(z)u(t-1)+\boldsymbol{B}u(t)$$

1.6.2.1 LQ 多变量控制

如果过程是渐近的，最小化准则可表示成：

$$J=e(t+1)^{\mathrm{T}}e(t+1)+u(t)^{\mathrm{T}}u(t)$$

其中

$$e(t+1)=r(t+1)-y(t+1)=[e_1(t+1)e_2(t+1)]^{\mathrm{T}}=\begin{pmatrix} r_1(t+1)-\hat{y}_1(t+1) \\ r_2(t+1)-\hat{y}_2(t+1) \end{pmatrix}$$

为预测误差向量。$\boldsymbol{R}$ 是对角线权重平方矩阵。

记

$$\Phi(t)=\boldsymbol{A}^*(z)y(t)+\boldsymbol{B}^*(z)u(t-1)$$

则最小化准则

$$J=[r(t+1)-\Phi(t)-\boldsymbol{B}u(t)]^{\mathrm{T}}[r(t+1)-\Phi(t)-\boldsymbol{B}u(t)]+u(t)^{T}Ru(t)$$

依赖如下所示控制向量的项

$$J_u=-\boldsymbol{M}^{\mathrm{T}}\boldsymbol{B}u(t)-[\boldsymbol{B}u(t)]^{\mathrm{T}}\boldsymbol{M}+[\boldsymbol{B}u(t)]^{\mathrm{T}}\boldsymbol{B}u(t)+u(t)^{\mathrm{T}}Ru(t)$$

其中 $M=r(t+1)-\Phi(t)$，等式两边同时对 $u(t)$ 求导得最优控制向量

$$u(t)=[R+\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B}]^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{M}=[R+\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B}]^{-1}\boldsymbol{B}^{\mathrm{T}}[r(t+1)-\boldsymbol{A}^*(z)y(t)-\boldsymbol{B}^*(z)u(t-1)]$$

1.6.2.2 LQI 多变量控制

如果过程中不含积分环节，同上述单变量过程，有

$$y(t+1)=[\boldsymbol{I}+\boldsymbol{A}^*(z)]y(t)-\boldsymbol{A}^*(z)y(t-1)+\boldsymbol{B}^*(z)\Delta u(t-1)+\boldsymbol{B}(z)\Delta u(t)$$

同样地：

$$J=e(t+1)^{\mathrm{T}}e(t+1)+\Delta u(t)^{\mathrm{T}}Ru(t)$$

得出最优控制量变化 $\Delta u(t)$ 向量：

$$\Delta u(t)=[R+\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B}]^{-1}\boldsymbol{B}^{\mathrm{T}}[r(t+1)-[\boldsymbol{I}+\boldsymbol{A}^*(z)]y(t)+\boldsymbol{A}^*(z)y(t-1)-\boldsymbol{B}^*(z)\Delta u(t-1)]$$

其中当有两个输出时，$\boldsymbol{I}$ 是 2 维单位矩阵。

1.6.3 应用举例

以一个两个输入、两个输出的汽热系统为例。

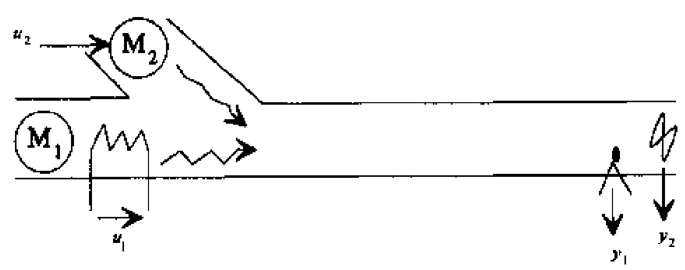


图 1-21 汽热系统原理图

图 1-21 的几点说明：

- 系统由 1 个 50cm 长的 PVC 管组成；
- 电机 M_1 和 M_2 在导管里产生空气流；
- 电机 M_1 通过热电阻提供一个稳定的加热空气流；
- 热空气输入 u_1 和冷空气 u_2 应用于电源印刷板，它没有在图中画出；
- 输出温度 y_1 和流量传感器 y_2 的调节电路板也同样没有画出。

用最小二乘法辨识得图 1-22 所示的汽热系统结构框图。

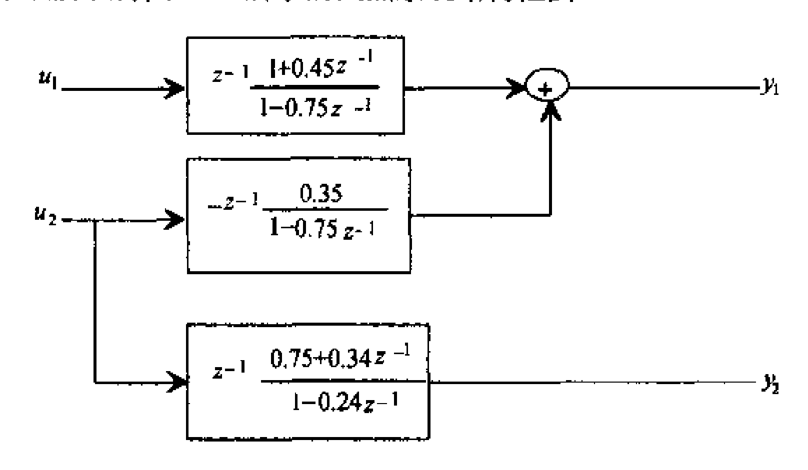


图 1-22 汽热系统结构框图

1.6.3.1 单变量热系统的 LQI 控制

以热流控制温度的 LQI 控制为例（见图 1-23）。

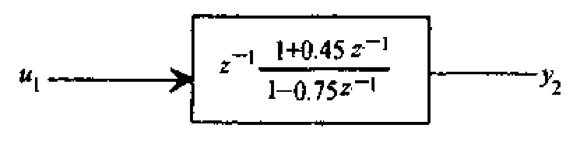


图 1-23 系统结构框图

预测输出模型为：

$$\hat{y}(t+1)=0.75y(t)+u(t)+0.45u(t-1)$$

则

$$A^*(z)=a_1=0.75, B^*(z)=b_2=0.45, B=b_1=1,$$

在 LQI 控制的预测模型中包含积分环节后，相应的控制变量为：

$$\Delta u(t)=\frac{b_1}{b_1^2+R}\left[r(t+1)-(1+a_1)y(t)+a_1 y(t-1)-b_2\Delta u(t-1)\right]$$

则控制量 $u(t)$ 为：

$$u(t)=\eta\frac{r(t+1)-[1+a_1-a_1 z^{-1}]y(t)}{(1-z^{-1})(1+\eta b_2 z^{-1})}$$

其中 $\eta=\dfrac{b_1}{b_1^2+R}$。

由此得闭环传递函数为：

$$\frac{y(t)}{r(t)}=\frac{\eta(b_1+b_2 z^{-1})}{1-z^{-1}[(1+a_1)(1-\eta b_1)-\eta b_2]+z^{-2}a_1(1-\eta b_1)}$$

闭环系统是具有单位静态增益的二阶系统，其阻尼系数 ξ 和自然振荡频率 $\Omega=\omega_0 T$ 由下列关系式给出：

$$(1+a_1)(1-\eta b_1)-\eta b_2=2\mathrm{e}^{-\xi\Omega}\cos(\Omega\sqrt{1-\xi^2})$$

$$a_1(1-\eta b_1)=\mathrm{e}^{-2\xi\Omega}$$

给定 $\xi=\dfrac{\sqrt{2}}{2}$。上述两式的算术比率由下面的等式给出：

$$\frac{(1+a_1)(1-\eta b_1)-\eta b_2}{a_1(1-\eta b_1)}=2\,\mathrm{e}^{\xi\Omega}\cos(\Omega\sqrt{1-\xi^2})$$

文件 w0TR.m 可画出上面等式中左右两个表达式的曲线，并显示出当横坐标 $x=\Omega=\eta\,\mathrm{b}_1$ 时两条曲线相交（见图 1-24）。

w0TR.m file

```
% Calculation of the w0T and R parameters of the LQI control
close all
clear all
% damping coefficient
z = sqrt(2)/2;

% process model params
a1 = 0.75;
b1 = 1;
b2 = 0.45;

% graph of the equality second term

x = 0:0.0001:0.9;
y = 2*exp(z*x).*cos(x*sqrt(1-z*z));
plot(x,y)
hold on

% graph of the equality first term
```

```
w = ((1+a1)*(1-x)-b2*x/b1)./(a1*(1-x));
plot(x,w,':')
axis([0 0.9 0 4])
xlabel('w0T normalized frequency and \eta b1 param')
title('search of the w0T frequency and of the \eta param')
gtext('(1+a1)(1-\eta b1)-\eta b2)/a1(1-\eta b1)')
gtext('2 e^{\zeta w0T} cos[w0T (1-\zeta^{2})^{1/2}]')
```

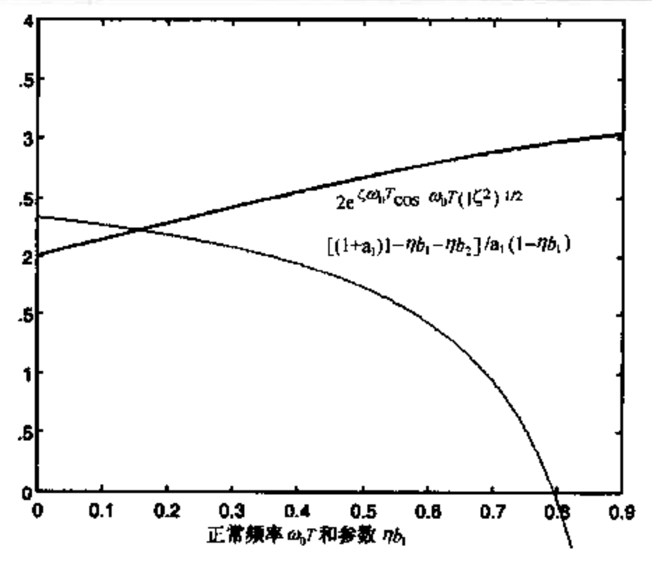


图 1-24　搜索频率ω_0和η值

$\Omega=\eta b_1$的值可由代表前面等式两端表达式曲线的横坐标得到，在这一点两条曲线以一定的精度相等。

这个横坐标为：

```
j = find(abs(y-w)<0.0001); w0T = x(j)
```

```
w0T =
    0.1572
```

权重系数R可由下面ηb_1表达式计算：

$$R=\frac{b_1^2(1-\eta b_1)}{\eta b_1}$$

```
R = b1*b1*(1-x(j))/x(j)

R =
   5.3613
```

在文件 lqi_mono.m 中，用上面得到的R系数计算控制量。

lqi_mono.m file

```
% LQI control of the monovariable thermal system

close all
clear all

% model process params
```

```
a1 = 0.75;
b1 = 1;
b2 = 0.45;

% R weighting coefficient

R = 5.3613;
eta = b1/(b1*b1+R);

% order signal

r = [3*ones(1,100) 5*ones(1,100) 3*ones(1,100)];

% signals initialization

du = 0;
y = r;
u = zeros(size(r));

for i = 3:length(r)-1
  % temperature output
  y(i) = a1*y(i-1)+b1*u(i-1)+b2*u(i-2);
  du = eta*(r(i+1)-(1+a1)*y(i)+a1*y(i-1)-b2*du);
  u(i) = u(i-1)+du;
end

% reference and temperature order graphs
figure(1)
plot(r)
hold on
h = plot(y);
set(h,'LineWidth',1.5)
title('Reference and output signals')
xlabel('samples')

% Control signal graph
figure(2)
stairs(u)
title('control signal')
xlabel('samples')
```

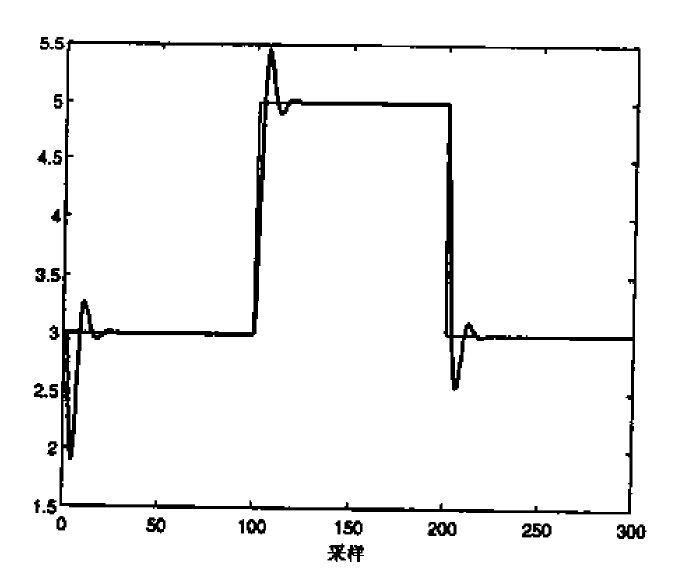


图 1-25　参考信号和输出信号

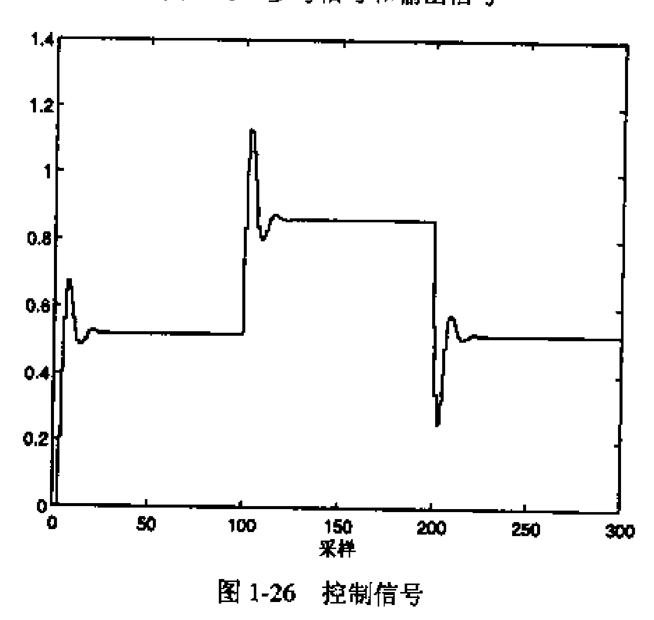


图 1-26　控制信号

如果 R 增加，响应时间增加并对控制信号有一点影响，由于有积分环节，稳态误差总是为 0。

1.6.3.2　多变量系统的 LQI 控制

预测模型矩阵为：

$$A^*(z)=\begin{bmatrix}0.75 & 0\\ 0 & 0.24\end{bmatrix}, B^*(z)=\begin{bmatrix}0.45 & 0\\ 0 & 0.34\end{bmatrix}, B=\begin{bmatrix}1 & 0.35\\ 0 & 0.75\end{bmatrix}$$

可推导得出最优控制量变化 $\Delta u(t)$ 向量为：

$$\Delta u(t)=[R+B^{\mathrm{T}}B]^{-1}B^{\mathrm{T}}[r(t+1)-[I+A^*(z)]y(t)+A^*(z)y(t-1)-B^*(z)\Delta u(t-1)]$$

在文件 lqi_mult.m 中任选权重系数实现了 LQI 多变量控制。

lqi_mult.m file

```
% LQI control of the multivariable thermal system
close all; clear all
```

```
% predictive model matrix
Aetoile = [0.75 0;0 0.24] ; Betoile = [0.45 0;0 0.34];
B = [1 -0.35;0 0.75];
% weighting matrix
R = [5.3613 0;0 5]; eta = inv(R + B'*B)*B';

% temperature and flow order signals
r1 = [ones(1,100) 5*ones(1,100) ones(1,100)];
r2 = [ones(1,50) 3*ones(1,150) ones(1,100)];
r = [r1;r2];

% signals initialization
du = [0;0]; y = r; u = r;

for i=3:length(r)-1
  % temperature output
  y(:,i) = Aetoile*y(:,i-1)+Betoile*u(:,i-2)+B*u(:,i-1);

  du = eta*(r(:,i+1)-(eye(size(Aetoile))+Aetoile)*y(:,i)+...
       Aetoile*y(:,i-1)-Betoile*du);
  u(:,i) = u(:,i-1)+du;
end

% temperature output graph
subplot(221)
plot(r1)
hold on
h = plot(y(1,:));
set(h,'LineWidth',1.5)
title('reference and output (temperature) signals')
xlabel('samples')

% heating control graph
subplot(222)
stairs(u(1,:))
title('heating control signal')
xlabel('samples')

% flow output graph
subplot(223)
plot(r2)
hold on
h = plot(y(2,:));
set(h,'LineWidth',1.5)
title('reference and output (flow) signals')
xlabel('samples')
axis([0 300 0.5 3.5])

% cold air control graph
subplot(224)
```

```
stairs(u(2,:))
title('control signal of the cold air flow')
xlabel('samples')
```

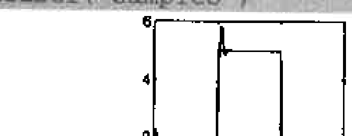
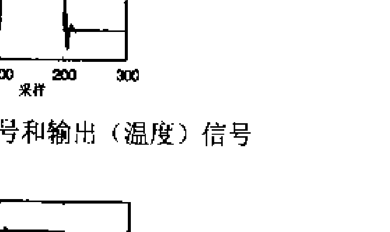



图 1-27　参考信号和输出（温度）信号

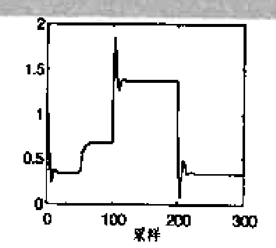


图 1-28　热流控制信号

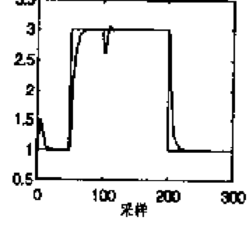


图 1-29　参考信号和输出（流）信号

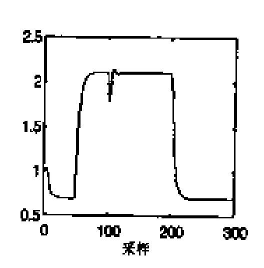


图 1-30　冷空气流控制信号

1.7　RST 控制

1.7.1　单变量系统

这个控制律基于一个多项式性能指标，它能够调整参考信号和输出信号（干扰）之间误差的动态性能。在下面一个单变量系统的例子中，如果 $P(z^{-1})$为调整抗干扰动态性能的多项式，则这个性能指标可由下式决定：

$$P(z)[r(t+1)-y(t+1)]=0$$

若选择一阶调整多项式

$$P(z)=1-0.5z^{-1}=1-p_1z^{-1},$$

则每个采样间隔的干扰降为原来的 1/2。以一个具有纯延时环节的一阶过程为例：

$$H(z^{-1})=\frac{B(z^{-1})}{A(z^{-1})}=\frac{z^{-1}(b_1+b_2z^{-1})}{1-a_1z^{-1}}$$

输出的预测模型为：

$$y(t+1)=a_1y(t)+b_1u(t)+b_2u(t-1)$$

如果过程是非渐近的，如前所述，在 LQI 控制中所用的方法来包含积分，由等式的分解组合得：

$$\hat{y}(t+1)=(1+a_1)y(t)-a_1y(t-1)+b_1\Delta u(t)+b_2\Delta u(t-1)$$

在本例中，具有一个一阶调整多项式的控制变量 $\Delta u(t)$不含积分环节，表达式为：

$$\Delta u(t)=\frac{1}{b_1}[r(t+1)-p_1r(t)-(1+a_1-p_1)y(t)+a_1y(t-1)-b_2\Delta u(t-1)$$

所以 t 时刻的控制量 $u(t)$为：

$$u(t)=u(t-1)+\Delta u(t)$$

如果模型是渐近的,直接得到控制量表达式为:

$$u(t)=\frac{1}{b_1}[r(t+1)-p_1 r(t)-(a_1-p_1)y(t)-b_2 u(t-1)]$$

一般的，同 LQI 控制，一个非渐近单变量系统有如下形式的一阶预测模型：

$$y(t+1)=\boldsymbol{A}^*(z)y(t-1)+\boldsymbol{B}^*(z)u(t-1)+\boldsymbol{B}u(t)$$

通过对这个表达式中两项的分解组合，得到一个新的模型：

$$y(t+1)=[1+\boldsymbol{A}^*(z)]y(t)-\boldsymbol{A}^*(z)y(t-1)+\boldsymbol{B}^*(z)\Delta u(t-1)+\boldsymbol{B}\Delta u(t)$$

由多项式性能指标得最优控制变化量：

$$\Delta u(t)=(\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B})^{-1}\boldsymbol{B}^{\mathrm{T}}[P(z)r(t+1)-[P(z)-1-\boldsymbol{A}^*(z)]y(t)+\boldsymbol{A}^*(z)y(t-1)-\boldsymbol{B}^*(z)u(t-1)$$

如果模型是渐近的，控制量为：

$$u(t)=(\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B})^{-1}\boldsymbol{B}^{\mathrm{T}}[P(z)r(t+1)-[P(z)-\boldsymbol{A}^*(z)]y(t)-\boldsymbol{B}^*(z)u(t-1)]$$

对于阶跃信号或方波脉冲，最好滤波后再作为决定跟踪动态性能的参考信号。一般选具有单位增益的一阶或二阶滤波器。一个这样的二阶参考滤波器如下：

$$H_{\mathrm{r}}(z^{-1})=\frac{1-\alpha_1-\alpha_2}{1-\alpha_1 z^{-1}-\alpha_2 z^{-2}}$$

若要选择具有极点α_1的一阶滤波器，只需使$\alpha_2=0$。对于没有陡峭前沿的阶次（正弦等），滤波是没有用的，这时取$\alpha_1=\alpha_2=0$。系数α_1、α_2根据下面的关系式决定无阻尼自然频率ω_0和阻尼系数ξ：

$$\alpha_1=2\mathrm{e}^{-\xi\omega_0 T}\cos(\omega_0 T\sqrt{1-\xi^2})$$

$$\alpha_2=-\mathrm{e}^{-2\xi\omega_0 T}$$

式中 T 是采样间隔。

所谓的 R、S、T 三支控制律如图 1-31 所示。

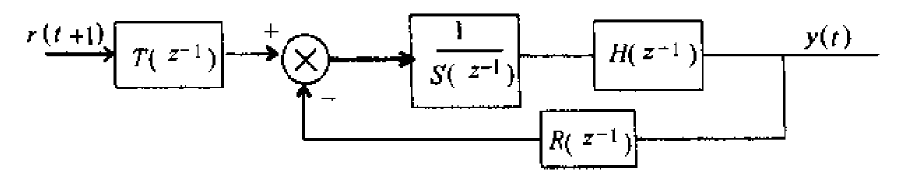


图 1-31　R、S、T 控制律结构框图

指令信号 $c(t)$滤波后，给出决定跟踪动态性能的参考信号 $r(t)$（见图 1-32)。

$c(t+1)$ → $H_{\mathrm{r}}(z^{-1})$ → $r(t+1)$

图 1-32　$c(t)$滤波后给出参考信号 $r(t)$

控制量由 3 个多项式 R、S、T 表示：

$$u(t)=\frac{1}{S(z^{-1})}[T(z^{-1})r(t+1)-R(z^{-1})y(t)]$$

在本例中有：

$$R(z^{-1}) = (1 + a_1 - p_1) - a_1 z^{-1}$$
$$S(z^{-1}) = b_1 + (b_2 - b_1)z^{-1} - b_2 z^{-2}$$
$$T(z^{-1}) = 1 - p_1 z^{-1}$$

1.7.2 多变量系统

以应用于 LQI 控制的汽热系统为例。这是一个二阶非渐近系统。为使具有抗干扰的动态性能，定义一个二阶多项式对角矩阵：

$$\boldsymbol{P}(z) = \begin{bmatrix} P_1(z) & 0 \\ 0 & P_2(z) \end{bmatrix}$$

每个多项式的次数定义了相应输出量的抗干扰的动态性能。

在一个非渐近系统中，多项式性能指标给定的控制变化向量如下：

$$\Delta u(t) = (\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B})^{-1}\boldsymbol{B}^{\mathrm{T}}[P(z)r(t+1) - [\boldsymbol{P}(z) - \boldsymbol{I} - \boldsymbol{A}^*(z)]y(t) + \boldsymbol{A}^*(z)]y(t-1) - \boldsymbol{B}^*(z)\Delta u(t-1)]$$

如果模型是渐近的，控制量 $u(t)$为：

$$u(t) = (\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B})^{-1}\boldsymbol{B}^{\mathrm{T}}[P(z)r(t+1) - [\boldsymbol{P}(z) - \boldsymbol{A}^*(z)]y(t) - \boldsymbol{B}^*(z)u(t-1)]$$

1.7.3 应用举例

1.7.3.1 温度的 RST 单变量控制

其传递函数见图 1-33。

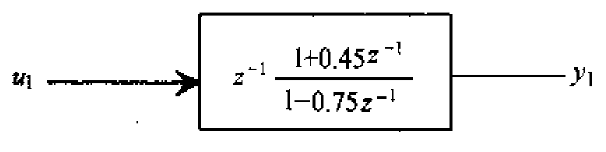


图 1-33 传递函数

积分预测模型为：

$$\hat{y}(t+1) = [1 + \boldsymbol{A}^*(z)]y(t) - \boldsymbol{A}^*(z)y(t-1) + \boldsymbol{B}^*(z)\Delta u(t-1) + b_1\Delta u(t)$$

其中

$$\boldsymbol{A}^*(z) = 0.75, \quad \boldsymbol{B}^*(z) = 0.45, \quad b_1 = 1。$$

选极点为 0.8 的一阶抗干扰多项式，即 $P(z^{-1})=1-0.8z^{-1}$。参考信号由指令信号通过一个具有单位静态增益的二阶滤波器产生。其中阻尼系数$\xi = \dfrac{\sqrt{2}}{2}$，衰减振荡频率$\omega_0 T=0.05\pi$ 。

在文件 rst_mono.m 中，以热流控制作为输入，温度变量作为输出，建立过程的 RST 单变量控制律。

rst_mono.m file

```
% LQI control of the du multivariable thermal system
close all; clear all

% Predictive model matrix
Aetoile = 0.75; Betoile = 0.45; B = 1;

% Regulation polynomial
```

```
P = 0.8;

% temperature order signal
c = [ones(1,100) 5*ones(1,100) ones(1,100)];

% reference model of the temperature
% second order of dzeta=0.7071
dzeta = 0.8; w0T=0.05*pi;
alfa1 = -2*exp(-dzeta*w0T)*cos(w0T*sqrt(1-dzeta^2));
alfa2 = exp(-2*dzeta*w0T);
r(1:2) = c(1:2);
 for i = 3:length(c)
     r(i) = (1+alfa1+alfa2)*c(i)-alfa1*r(i-1)-alfa2*r(i-2);
 end

% signals initialization
du = 0; y = r; u = zeros(size(r));
P_I_A = P-1-Aetoile;
for i = 3:length(r)-1
  % temperature output
  y(i) = Aetoile*y(i-1)+B*u(:,i-1)+Betoile*u(i-2);
  du = inv(B'*B)*B'*(r(i+1)- P*r(i)+P_I_A*y(i)+...
       Aetoile*y(i-1)-Betoile*du);
  u(i) = u(i-1)+du;
  y(i) = y(i) - 2* (i == 110);
end

% temperature output graph
figure(1)
plot(c), hold on
plot(r,':')
h = plot(y);
set(h,'LineWidth',1.5)
title('order, reference and output (temperature) signals')
xlabel('samples'), axis([0 300 0 6])

% heating control graph
figure(2)
stairs(u)
title('heating control signal')
xlabel('samples')
```

在 t=110 的离散时刻，加入干扰信号 2。由于加入了极点为 0.8 的一阶多项式，所以排除了干扰和初始条件的影响，而指令信号跟踪通过一个二阶系统滤波后发生。

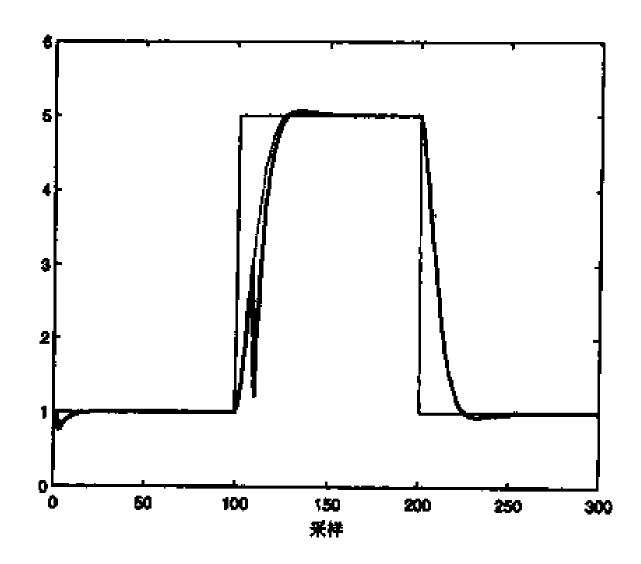


图 1-34　阶数、参考信号和输出（温度）信号

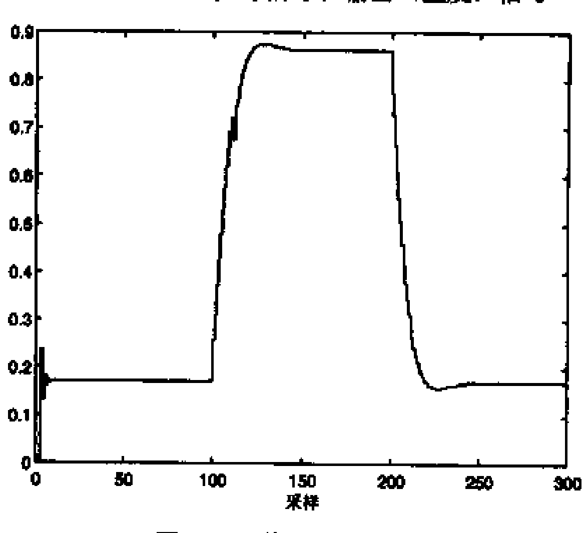


图 1-35　热流控制信号

1.7.3.2　汽热过程的 RST 多变量控制

其模型矩阵为：

$$A^{*}(z)=\begin{bmatrix}0.75 & 0\\ 0 & 0.24\end{bmatrix},\quad B^{*}(z)=\begin{bmatrix}0.45 & 0\\ 0 & 0.34\end{bmatrix},\quad B=\begin{bmatrix}1 & 0.35\\ 0 & 0.75\end{bmatrix}$$

控制律为：

$$\Delta u(t)=(B^{\mathrm{T}}B)^{-1}B^{\mathrm{T}}[P(z)r(t+1)-[I+A^{*}(z)]y(t)+A^{*}(z)]y(t-1)-B^{*}(z)\Delta u(t-1)]$$

注意：上式与 R=0 时的 LQ 控制一样。

rst_mult.m file

```
% LQI control of the multivariable thermal system
close all; clear all

% predictive model matrix

Aetoile = [0.75 0;0 0.24];
```

```
Betoile = [0.45 0;0 0.34];
B = [1 -0.35;0 0.75];
% regulation matrix
P = [0.8 0;0 0.5];

% temperature and flow order signals

c1 = [ones(1,100) 5*ones(1,100) ones(1,100)];
c2 = [ones(1,50) 3*ones(1,150) ones(1,100)];

% temperature and flow reference models
% first order model for the temperature

pole = 0.8;
r1 = c1(1);

 for i = 2:length(c1)
    r1(i) = (1-pole)*c1(i)+pole*r1(i-1);
 end


% second order model for the flow
% reference model of the position order
% second order of dzeta=0.7071

dzeta = 0.8; %sqrt(2)/2;
w0T = 0.05*pi;
alfa1 = -2*exp(-dzeta*w0T)*cos(w0T*sqrt(1-dzeta^2));
alfa2 = exp(-2*dzeta*w0T);
r2(1:2) = c2(1:2);

 for i = 3:length(c2)
    r2(i) = (1+alfa1+alfa2)*c2(i)-alfa1*r2(i-1)-alfa2*r2(i-2);
 end

r = [r1;r2];

% signals initialization
du = [0;0];
y = r;
u = r;

% (P I - A) control matrix
P_I_A = P-eye(size(P))-Aetoile;

for i = 3:length(r)-1
  % temperature output
  y(:,i) = Aetoile*y(:,i-1)+B*u(:,i-2)+Betoile*u(:,i-1);
du = inv(B'*B)*B'*(r(:,i+1)- P*r(:,i)+P_I_A*y(:,i)+...
    Aetoile*y(:,i-1)-Betoile*du);
u(:,i) = u(:,i-1)+du;
```

```
  % temperature disturbance
  y(1,i) = y(1,i) - 2*(i==120);
end

% temperature output graph
subplot(121), hold on
plot(c1), plot(r1,':')
h = plot(y(1,:)) ; set(h,'LineWidth',1.5)
title('order, reference and temperature output')
xlabel('samples'), axis([0 300 0 6])

% heating control graph
subplot(122), stairs(u(1,:))
title('heating control signal')
xlabel('samples')

% flow output graph
figure(2), subplot(121), hold on
plot(c2), plot(r2)
h = plot(y(2,:)); set(h,'LineWidth',1.5)
title('reference and flow order output')
xlabel('samples')

% cold air control graph
subplot(122), stairs(u(2,:))
title('control signal of the cold air flow')
xlabel('samples')
```

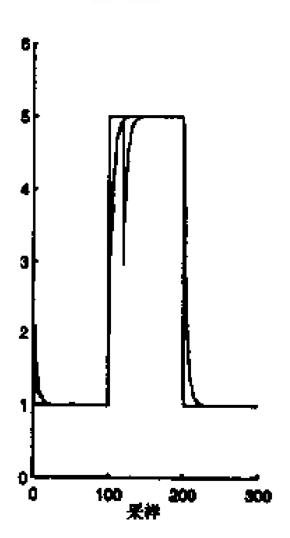


图 1-36　阶数、参考信号和温度输出信号

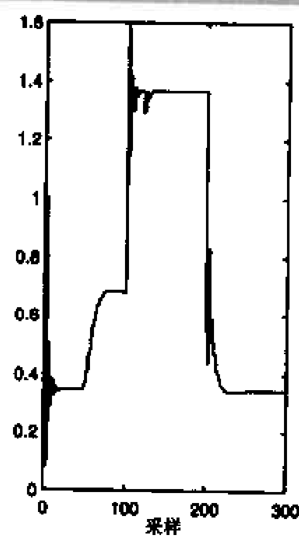


图 1-37　热流控制信号

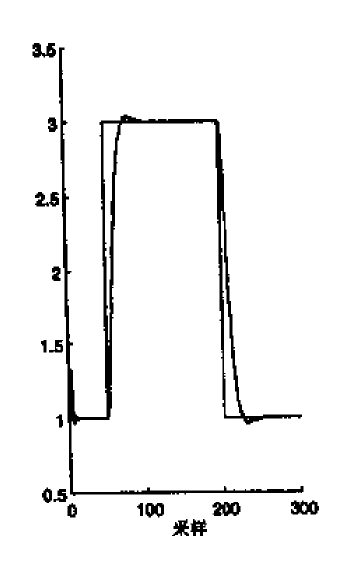


图 1-38　参考信号和流输出信号

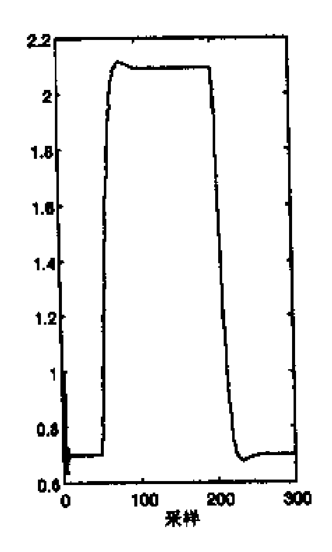


图 1-39　冷空气流控制信号

第 2 章　连续系统和离散系统的状态空间描述

2.1　连续系统的状态空间描述

2.1.1　*启发式方法*

以直流电动机为例，其传递函数如下：

- J 为转子惯性力矩；
- r、L 为电枢阻抗和感应系数；
- f 为转子的液压摩擦系数；
- k 为耦合系数。

$$E(p)=U(p)-rI(p)-LpI(p)$$

$$E(p)=k\,\Omega$$

$$Cm(p)=kI(p)=Jp\,\Omega(p)+F\,\Omega(p)$$

经过计算得

$$M(p)=\frac{\Omega(p)}{U(p)}=\frac{k}{k^2+(r+Lp)(Jp+F)}$$

即

$$M(p)=\frac{b_0}{p^2+a_1p+a_0}$$

其中

$$\begin{cases}b_0=\dfrac{k}{LJ}\\[2mm] a_0=\dfrac{k^2+rF}{LJ}\\[2mm] a_1=\dfrac{F}{J}+\dfrac{r}{L}\end{cases}$$

<u>*注意*</u>

$Y(p)=\Omega(p)$：过程输出；

$U(p)$　　：过程输入。

$$b_0U(p)=(p^2+a_1p+a_0)Y(p)$$

假定初始条件为零，可以导出如下微分方程：

$$\frac{\mathrm{d}^2y(t)}{\mathrm{d}t^2}+a_1\frac{\mathrm{d}y(t)}{\mathrm{d}(t)}+a_0y(t)=b_0u(t)$$

这个系统是二阶的，需要两个状态变量 x_1 和 x_2（见图 2-1）。

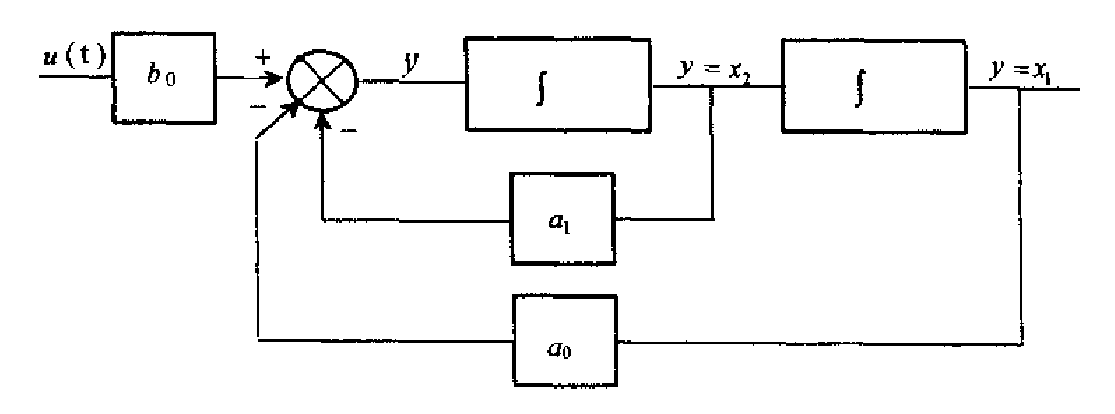


图 2-1　系统结构框图

状态变量代表积分器的输出，即

$$x_1(t) = y(t); \qquad x_2(t) = \dot{x}_2(t)$$

$$\dot{x}_2(t) = -a_1 x_2(t) - a_0 x_1(t) + b_0 u(t)$$

矩阵形式为：

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -a_0 & -a_1 \end{bmatrix} \times \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ b_0 \end{bmatrix} \times u(t)$$

$$\dot{X}(t) = \boldsymbol{A}X(t) + \boldsymbol{B}u(t)$$

$$y(t) = [1 \quad 0] \times \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

$$y(t) = \boldsymbol{C}X(t)$$

2.1.2　广义状态空间描述

状态空间描述法的广义形式如下：

状态方程：

$$\dot{X}(t) = \boldsymbol{A}X(t) + \boldsymbol{B}u(t)$$

输出方程：

$$Y(t) = \boldsymbol{C}X(t) + \boldsymbol{D}U(t)$$

初始条件：X_0

<u>状态矩阵</u>

下面给出状态方程的拉普拉斯变换：

$$pX(p) - X_0 = \boldsymbol{A}X(p) + \boldsymbol{B}U(p)$$

$$(p\boldsymbol{I} - \boldsymbol{A})X(p) = X_0 + \boldsymbol{B}U(p)$$

$$X(p) = (p\boldsymbol{I} - \boldsymbol{A})^{-1} X_0 + (p\boldsymbol{I} - \boldsymbol{A})^{-1} \boldsymbol{B}U(p)$$

<u>*注意*</u>　　$(p\boldsymbol{I} - \boldsymbol{A})^{-1} = L[\mathrm{e}^{At}]$

状态方程的时域表示如下：

$$X(t) = \mathrm{e}^{A(t-t_0)} X(t_0) + \int_{t_0}^{t} \mathrm{e}^{A(t-\tau)} \boldsymbol{B}U(\tau)\mathrm{d}\tau$$

$$Y(t) = \boldsymbol{C}X(t) + \boldsymbol{D}U(t) \qquad t \geqslant t_0$$

2.2 离散系统的状态空间描述

2.2.1 启发式方法

以一个忽略电枢感应的简化直流电动机模型为例。把相对于电动机轴线的转角作为过程输出：

$$M(p)=\frac{\theta(p)}{U(p)}=\frac{K}{p(1+\tau p)}$$

其中

$$\begin{cases} K=\dfrac{k}{k^2+rF} \\ \tau=\dfrac{rJ}{k^2+rF} \end{cases}$$

即

$$M(p)=\frac{b_0}{p^2+a_1 p}$$

其中

$$\begin{cases} b_0=\dfrac{K}{\tau} \\ a_1=\dfrac{1}{\tau} \end{cases}$$

系统结构框图如图 2-2 所示。

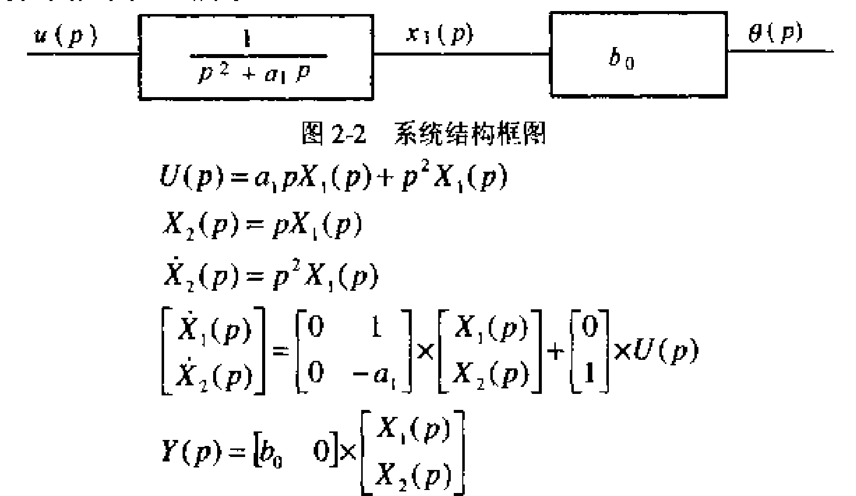


图 2-2 系统结构框图

$$U(p)=a_1 pX_1(p)+p^2X_1(p)$$

$$X_2(p)=pX_1(p)$$

$$\dot{X}_2(p)=p^2X_1(p)$$

$$\begin{bmatrix}\dot{X}_1(p)\\ \dot{X}_2(p)\end{bmatrix}=\begin{bmatrix}0 & 1\\ 0 & -a_1\end{bmatrix}\times\begin{bmatrix}X_1(p)\\ X_2(p)\end{bmatrix}+\begin{bmatrix}0\\ 1\end{bmatrix}\times U(p)$$

$$Y(p)=\begin{bmatrix}b_0 & 0\end{bmatrix}\times\begin{bmatrix}X_1(p)\\ X_2(p)\end{bmatrix}$$

这个过程的另一种状态空间描述方程为：

$$X_1(p)=Y(p)=\theta(p)$$

$$X_2(p)=pX_1(p)$$

$$\begin{bmatrix}\dot{X}_1(p)\\ \dot{X}_2(p)\end{bmatrix}=\begin{bmatrix}0 & 1\\ 0 & -a_1\end{bmatrix}\times\begin{bmatrix}X_1(p)\\ X_2(p)\end{bmatrix}+\begin{bmatrix}0\\ b_0\end{bmatrix}\times U(p)$$

$$Y(p)=\begin{bmatrix}1 & 0\end{bmatrix}\times\begin{bmatrix}X_1(p)\\ X_2(p)\end{bmatrix}$$

矩阵 A_d、B_d、C_d 和 D_d 是连续系统矩阵 A、B、C 和 D 在相应的离散系统中对应的矩阵。

以 T_s 作为系统离散的采样周期，kT_s 时刻过程参量公式如下：

$$\begin{cases} X(k+1) = A_d X(k) + B_d U(k) \\ Y(k) = C_d X(k) + D_d U(k) \end{cases}$$

已知 t_0 时刻连续系统的状态变量，则 t_0+T_s 时刻的状态变量为：

$$X(t_0+T_s) = e^{AT_s} X(t_0) + \int_{t_0}^{t_0+T_s} e^{A(t_0+T_s-\tau)} BU(\tau)d\tau$$

用 kT_s 代替 t_0 并假定 T_s 为瞬时值，可以得到：

$$X[(k+1)T_s] = X(k+1)$$

$$X(k+1)) = e^{AT_s} X(k) + \int_k^{k+1} e^{A[(k+1)T_s-\tau]} BU(\tau)d\tau$$

设 $t' = \tau - kT_s$，则输入 $u(t) = u(k)$ 在 T_s 内不变为常量，则有：

$$X(k+1)) = e^{AT_s} X(k) + \int_0^{T_s} e^{A(T_s-t')} BU(k)dt'$$

$$X(k+1)) = e^{AT_s} X(k) + \int_0^{T_s} e^{A(T_s-t')} dt' BU(k)$$

由此证明：

$$\begin{cases} A_d = e^{AT_s} \\ B_d = \int_0^{T_s} e^{A(T_s-t')} dt' B \\ C_d = C \\ D_d = D \end{cases}$$

2.2.2 应用

以位置为输出的简化电动机离散状态描述如下：

$$A_d = e^{AT_s}，其中 \quad A = \begin{bmatrix} 0 & 1 \\ 0 & -a_1 \end{bmatrix}$$

$$L[e^{AT_s}] = (pI - A)^{-1}$$

即

$$A_d = L^{-1} \begin{bmatrix} p & -1 \\ 0 & p+a_1 \end{bmatrix}^{-1}$$

矩阵逆变换得

$$A_d = L^{-1} \begin{bmatrix} \dfrac{1}{p} & \dfrac{1}{p(p+a_1)} \\ 0 & \dfrac{1}{p+a_1} \end{bmatrix}$$

所以

$$A_d = \begin{bmatrix} 1 & \dfrac{1-e^{-a_1T_s}}{a_1} \\ 0 & e^{-a_1(T_s-t')} \end{bmatrix}$$

并且

$$B_d = \int_0^{T_s} e^{A(T_s-t')}dt'B = \int_0^{T_s} \begin{bmatrix} 1 & \dfrac{1-e^{-a_1T_s}}{a_1} \\ 0 & e^{-a_1(T_s-t')} \end{bmatrix} dt' \times \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$B_d = \int_0^{T_s} \begin{bmatrix} 1 & \dfrac{1-e^{-a_1T_s}}{a_1} \\ 0 & e^{-a_1(T_s-t')} \end{bmatrix} dt'$$

即

$$B_d = \begin{bmatrix} \dfrac{T_s}{a_1} - \dfrac{1+e^{-a_1T_s}}{a_1^2} \\ \dfrac{1-e^{-a_1T_s}}{a_1} \end{bmatrix}$$

输出方程系数矩阵为：

$$C_d = C = [b_0 \quad 0] \qquad D_d = D = 0$$

2.3 可控性和可观测性

2.3.1 可控性

定义

给定一个状态向量 $X(t_i)$，如果在控制量 $u(t)$的控制下，从任意状态最终都可以到达一个最终状态 $X(t_f)$，那么就称系统可控。

定理

给定一个 n 阶系统：

连续：$\dot{X}(t) = AX(t) + BU(t)$

离散：$X(k+1) = A_dX(k) + B_dU(k)$

如果可控矩阵的秩等于 n，系统可控。

可控矩阵

连续：$\text{Com} = [B \quad AB \quad \cdots \quad A^{n-1}B]$

离散：$\text{Com} = [B_d \quad A_dB_d \quad \cdots \quad A_d^{n-1}B_d]$

2.3.2 可观测性

定理

由以下状态方程和观察方程定义一个 n 阶系统：

连续系统：

$$\dot{X}(t) = AX(t) + BU(t) \qquad Y(t) = CX(t) + DU(t)$$

离散系统：

$$X(k+1)=A_dX(k)+B_dU(k)\qquad Y(k)=C_dX(k)+D_dU(k)$$

如果输出矩阵的秩等于 n，则系统可观测。

观测矩阵

连续系统：

$$\text{Obs}=\begin{bmatrix}C & CA & \cdots & CA^{n-1}\end{bmatrix}^{\mathrm{T}}$$

离散系统：

$$\text{Obs}=\begin{bmatrix}C_d & C_dA_d & \cdots & C_dA_d^{\,n-1}\end{bmatrix}^{\mathrm{T}}$$

2.4 离散动态系统的状态重构

如果过程可观测，就可以用输入量 $u(t)$和输出量 $y(t)$以及矩阵 A_d和 B_d重构其状态变量。状态重构的好处有很多，在状态变量的测量非常困难、不可能或者太昂贵时可以用状态重构法得到状态变量。用传感器测量状态变量并不是非常可靠，或者会产生很强的噪声。在此讲述 2 种状态重构器或观测器：

- 确定性过程观测器，或 Luenberger 观测器；
- 随机过程观测器，或卡尔曼预测器。

2.4.1 确定性过程的闭环估计

已知A_d、B_d和C_d是过程或系统的矩阵，则

$$X(k+1)=A_dX(k)+B_dU(k)$$
$$Y(k)=C_dX(k)$$

图 2-3 确定性过程的闭环估计

◆ 状态重构器（见图 2-4）

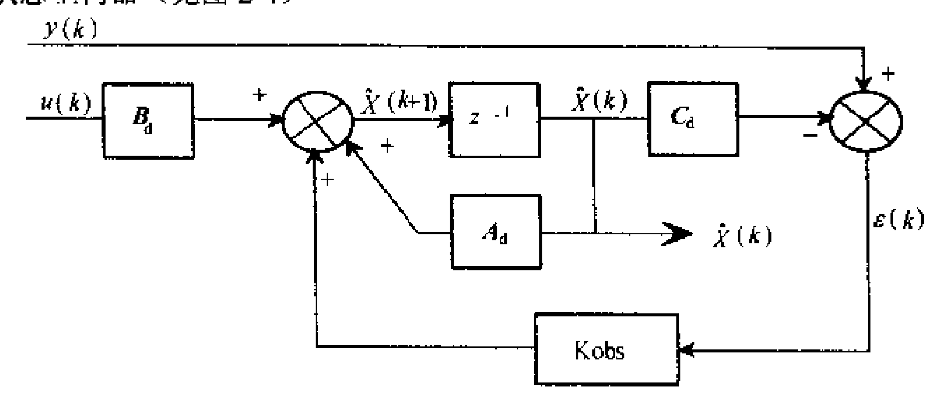


图 2-4 状态重构器

以 $\hat{X}(k+1)$作为输出，构造状态重构预测器。

◆ 预测方程

$$\hat{X}(k+1)=A_d\hat{X}(k)+B_du(k)+Kobs[y(k)-\hat{y}(k)]$$
$$\hat{X}(k+1)=A_d\hat{X}(k)+B_du(k)+Kobs[y(k)-C_d\hat{X}(k)]$$

即

$$\hat{X}(k+1)=(A_d-KobsC_d)\hat{X}(k)+B_du(k)+Kobsy(k)$$

Kobs 列矩阵是一个增益矩阵。必须选择它达到估计快速收敛同时保持系统稳定的目的。

◆ 稳定性准则

考虑

$$\tilde{X}(k)=\hat{X}(k)-X(k)$$
$$\tilde{X}(k+1)=(A_d-KobsC_d)\hat{X}(k)+B_du(k)+Knobs\ y(k)-A_dX(k)-B_du(k)$$

即

$$\tilde{X}(k+1)=(A_d-KobsC_d)\tilde{X}(k)$$

选择 *Kobs* 使得误差演化矩阵（A_d－ $Kobs\ C_d$）特征值的模小于 1，这也决定了估计误差的动态特性。误差极点是以下特征方程的解：

$$\det\lfloor zI-A-KobsC_d\rfloor=0$$

应该使它们尽可能靠近原点。

<u>注意</u>　*Kobs*为常量增益矩阵，能够特征化误差动态特性，但观测器并不是最优。当 *Kobs*=*Kobs*(*t*)随时间变化而变化时，观测器有最优解。

2.5　状态反馈控制

增加状态向量反馈控制来校正过程的动态表现。

<u>原理</u>（见图 2-5）

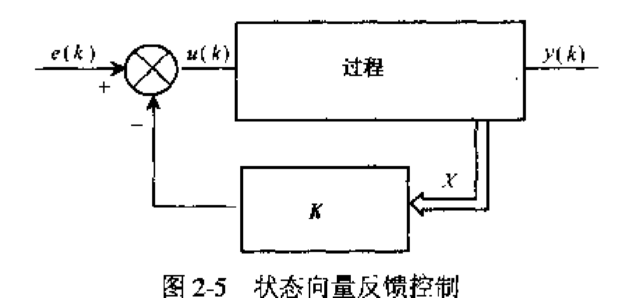


图 2-5　状态向量反馈控制

$$u(k)=e(k)-KX(k)$$
$$X(k+1)=A_dX(k)+B_du(k)=(A_d-KB_d)X(k)+B_de(k)$$

即

$$\begin{cases}X(k+1)=(A_d-KB_d)X(k)+B_de(k)\\Y(k)=C_dX(k)\end{cases}$$

现在状态反馈控制对过程动态的校正就归结为矩阵（A_d-KB_d）的特征方程的一个函数。以 α_i 为校正系统极点，计算 K 向量：

$$\det[A_d - KB_d] = z^n + \alpha_{n-1}z^{n-1} + \cdots + \alpha_0$$

为便于计算，写出 A_d 的伴随矩阵形式，也称为可控标准形式：

$$A_{dc} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-1} \end{bmatrix} \quad B_{dc} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

方法

计算变换矩阵 W 如下：

$$\begin{cases} W^{n-1} = A_d W^n + a_{n-1}B_d \\ W^n = B_d \end{cases}$$

应用如下变换得到描述系统的伴随矩阵形式：

$$A_{dc} = W^{-1}A_d W$$

$$B_{dc} = W^{-1}B_d$$

记 $K=[k_0 \quad k_1 \cdots k_{n-1}]$，得到：

$$[A_{dc} - B_{dc}K] = \begin{bmatrix} 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots & 0 \\ 0 & 0 & \cdots & 0 & 1 \\ -(a_0+k_0) & -(a_1+k_1) & \cdots & -(a_{n-2}+k_{n-2}) & -(a_{n-1}+k_{n-1}) \end{bmatrix}$$

其特征方程为：

$$\det[zI - (A_{dc} - B_{dc}K)] = z^n + (a_{n-1}+k_{n-1})z^{n-1} + \cdots + (a_0+k_0)$$

由此确定：

$$\alpha_i = a_i + k_i$$

计算得：

$$K_c = KW^{-1}$$

2.6 例子

2.6.1 有积分环节过程的状态反馈控制系统

在这个例子中，用状态变量反馈来控制本章第 2 节讲到的直流电动机位置输出。

例子文件是 state_var1.m。

首先通过过程的开环指数响应，构造状态空间描述以便建立数学模型。

state_var1.m file

```
% Position control device of the direct current motor

% Direct current motor characteristics
r=3; L=0.001;
f=0.002; J=0.01;
```

```
k=1.2;

K=k/(k^2+r*f);
T=r*J/(k^2^+r*f);

% Motor transfer function, position output
num=K;
den=[T 1 0];
sys=tf(num,den)
Transfer function:
   0.8299
-----------------
5.023 s^2 + s
```

初始化状态空间描述矩阵，验证上面计算所得的传递函数。

```
% A,B,C,D model verification
A=[0 1; 0 -1/T];
B=[0 K/T]';
C=[1 0];
D=0;
sys_ss=ss(A,B,C,D);
tf(sys_ss)

Transfer function:
   0.1652
-----------------
s^2 +0.1991 s
```

结果与前面计算所得的传递函数等价。

图 2-6 表示直流电机的阶跃响应和位置输出。

```
% Indicial response
figure(1);
[y,t,x]=step(sys_ss);
plot(t,y),grid;
title('Indicial response of the direct current motor');
xlabel('Time');
ylabel('Position in rad');
```

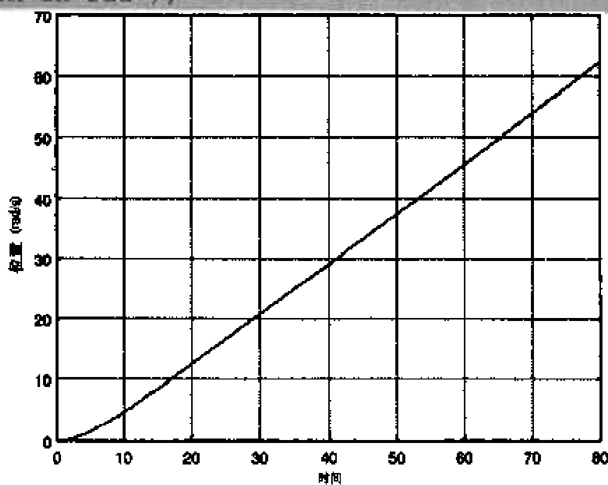


图 2-6　直流电机的阶跃响应曲线

由图可以看出暂态过后，对应于定速旋转的位置输出是线性变化的。

用 ssdata 函数可以从过程的传递函数得到其状态空间描述。

```
% State representation from the transfer function
[A1,B1,C1,D1]=ssdata(sys)
A1 =
   - 0.1991   0
0.5000   0
B1 =
0.5000
0
C1 =
        0    0.6608
D1=
    0
```

计算过程的另一种状态空间描述。用 obsv 函数计算输出矩阵，用 rank 函数计算它的秩。

```
% Observability verification
obs=obsv(sys_ss);
rang=rank(obs)

rang=
    2
```

过程是可观测的。Ctrb 函数用来判断过程的可控性。

```
% Steerability verification
com=ctrb(sys_ss);
rang=rank(com)

rang=
    2
```

可控矩阵的秩等于过程的阶数，所以过程是可控的。

用 canon 函数计算状态空间描述的伴随矩阵形式。

```
% Writing in companion form
sys_comp=canon(sys_ss,'companion');
[Ac,Bc,Cc,Dc]=ssdata(sys_comp)
Ac =
0        0
1. 0000  -0. 1991
Bc =
1
0
Cc =
        0      0. 1652
Dc =
    0
```

现在用极点配置法计算状态反馈矩阵 $\boldsymbol{K}$，来控制电机轴线位置。

$$\dot{x} = \boldsymbol{A}x + \boldsymbol{B}u$$

$$u = -\boldsymbol{K}x$$

配置极点使闭环过程具有理想的动态性能。其稳定性和暂态特征由矩阵（$\boldsymbol{A}-\boldsymbol{BK}$）的

特征值决定。为得到一个衰减振荡二阶过程响应，配置极点为一对复极点：

$$p_{1,2} = -z\omega_n \pm \mathrm{j}\omega_n\sqrt{1-z^2}$$

其中

$$\begin{cases} \omega_n = 10\times\dfrac{2\pi}{T} \\ z = \dfrac{\sqrt{2}}{2} \end{cases}$$

下面计算复极点并绘制闭环过程的理想指数响应曲线。

```
% Poles choice of the closed-loop system
wn=20*pi/T;
z=sqrt(2)/2;
p1=-z*wn+(wn*sqrt(1-z^2))*i;
p2=-z*wn-(wn*sqrt(1-z^2))*i;
sysc=zpk([],[p1 p2],abs(p1*p2));
[yc,tc]=step(sysc);
figure(2);
plot(tc,yc),grid;
title('Searched for indicial response');
xlabel('Time');
ylabel('Position in rad');
p1=
  -8.8443 + 8.8443i
p2=
  -8.8443 - 8.8443i
```

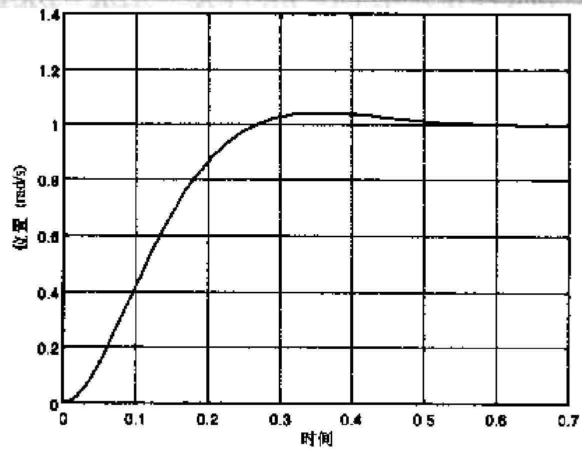


图 2-7　闭环过程的理想指数响应曲线

对于状态反馈的伺服控制，所选状态向量为要控制的输出量的函数，这里 $y(t)=\theta(t)=x_1(t)$（见图 2-8）。

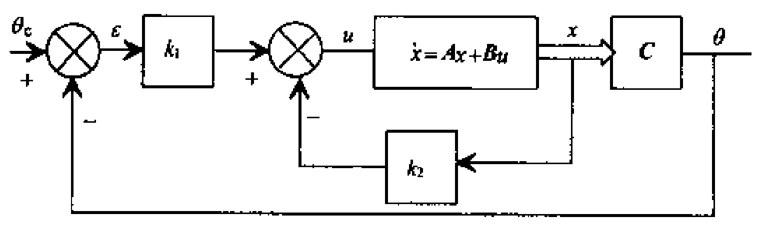


图 2-8　状态反馈的伺服控制

状态反馈向量

$$K=[k_1 \quad k_2]$$

过程控制

$$u=-Kx+k_1x_1+k_1(\theta_c-y)$$

选择状态向量使 $y=x_1$，对于闭环系统，得到：

$$u=-Kx+k_1\theta_c$$
$$\dot{x}=(A-BK)x+Bk_1\theta_c$$

◆ 用 Ackerman 公式计算 K

理想特征方程如下：

$$|pi-A+BK|=p^n+\alpha_{n-1}p^{n-1}+\cdots+\alpha_0$$

注意到：

$$\phi(A)=A^n+\alpha_{n-1}A^{n-1}+\cdots+\alpha_0 I$$
$$A=A-BK$$

由 Cayley-Hamilton 定理可得：

$$\phi(A)=A^n+\alpha_{n-1}A^{n-1}+\cdots+\alpha_0 I=0$$

为推导出 Ackerman 公式，将过程阶数 n=2 代入。由此证明得：

$$\begin{cases}\hat{A}=A-BK\\ \hat{A}=(A-BK)^2=A^2-ABK-ABK+(BK)^2=A^2-ABK-BK\hat{A}\end{cases}$$

即

$$\phi(\hat{A})=\alpha_0 I+\alpha_1(A-BK)+A^2-ABK-BK\hat{A}=0$$
$$\phi(\hat{A})=\phi(A)+\alpha_1 BK-ABK-BK\hat{A}=0$$

所以

$$\phi(A)=B(\alpha_1 K+K\hat{A})+ABK$$

写成矩阵形式

$$\phi(A)=[B \quad AB]\begin{bmatrix}\alpha_1 K+K\hat{A}\\ K\end{bmatrix}$$

由此等式计算出

$$K=[0 \quad 1][B \quad AB]^{-1}\phi(A)$$

可以看出矩阵[B AB]实际上是可控过程的可控矩阵，经推理归纳得出 n 阶 Ackerman 等式如下：

$$K=[0 \quad 1]\text{com}^{-1}\phi(A)$$

下面应用 polyvalm 函数，由理想极点的特征多项式和 $\phi(\hat{A})$ 计算 K。

```
% Searched characteristic polynomial
p=poly([p1 p2]);

% phi characteristic polynomial
phi=polyvalm(p,A)

% state return vector calculation
```

```
% Ackermann formula utilization
K=[0 1]*(inv(com))*phi

p =
   1. 0000   17. 6885   156. 4416
phi =
  156. 416   17. 4894
        0  152. 9600
k=
 946. 9848  105.8685
```

用前面描述的公式表示闭环过程的状态。

```
% State representation of the closed-loop system
Ab=A-B*K
Bb=B*K(1,1)
Cb=C
Db=D

Ab =
          0     1. 0000
   -156. 4416  -17. 6885

Bb =
        0
   156.4416
Cb =
1    0
Db =
    0
% Closed-loop system indicial response
figure(3);
[y,x,t]=step(Ab,Bb,Cb,Db);
plot(t,y),grid;
title('Closed-loop system indicial response');
xlabel('Time');
ylabel('Position in rad');
```

极点配置后的闭环过程的指数响应与理想的指数响应一致（见图 2-9）。

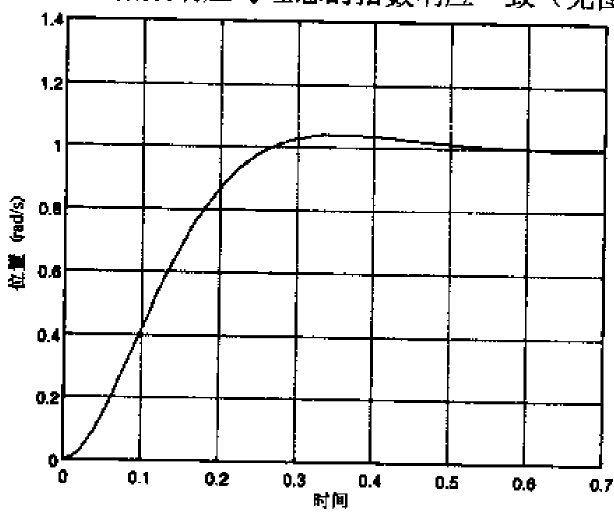


图 2-9　闭环过程的阶跃响应曲线

从状态变量的阶跃响应可以看出，状态 $x_1(t)$代表输出$\theta(t)$，$x_2(t)$是其导数，如图 2-10 所示。

```
figure(4);
plot(t,x),grid;
title('Indicial response of x1(t)=y(t) and of x2(t)=x1''(t)');
xlabel('Time');
ylabel('x1(t) x2(t)');
gtext('x1(t)'),gtext('x2(t)');
```

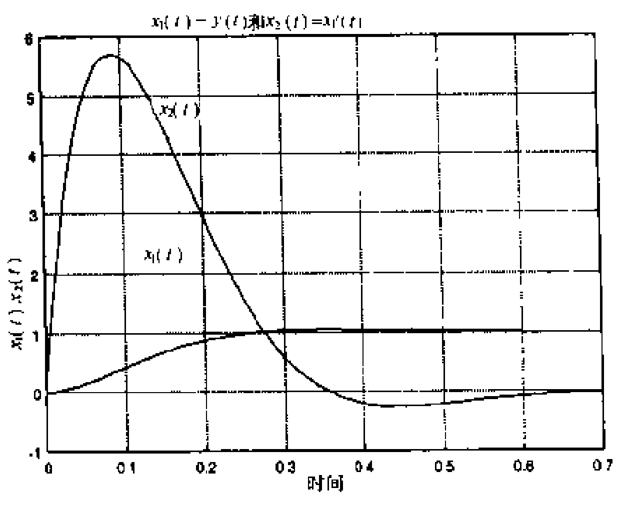


图 2-10 状态变量的阶跃响应

2.6.2 无积分环节过程的状态反馈控制系统

在这个例子中用状态变量控制本章第 1 节提到过的直流电机速度。例子文件为 **state_var2.m**。首先通过过程的开环阶跃响应构造状态空间描述，以便建立数学模型。

state_var2.m file

```
% Speed servo-control of the direct current motor

% direct current motor characteristics
r=3; L=0.001;f=0.002; J=0.01;k=1.2;
a0=(k^2+r*f)/(L*J);a1=f/J+r/L;b0=k/(L*J);

% Motor transfer function, position output
num=b0;
den=[1 a1 a0];
sys=tf(num,den)

Transfer  function :
1. 2e005
------------------------------
s^2 + 3000 s + 1.446e005
% A,B,C,D model verification
A=[0 1; -a0 -a1];
B=[0 b0]';
C=[1 0];
D=0;
sys_ss=ss(A,B,C,D);
```

```
tf(sys_ss)

Transfer function :
1. 2e005
-------------------------------
s^2 + 3000 s + 1.446e005
```

两种方法得到的传递函数是一致的。

开环电机的波特图和阶跃响应分别如图 2-11、图 2-12 所示。

```
% Bode diagrams
figure(1);
bode(sys_ss);

% Indicial response
figure(11);
[y,t,x]=step(sys_ss);
plot(t,y),grid;
title('Indicial response of the direct current motor');
xlabel('Time');ylabel('Speed in rad/s');
```

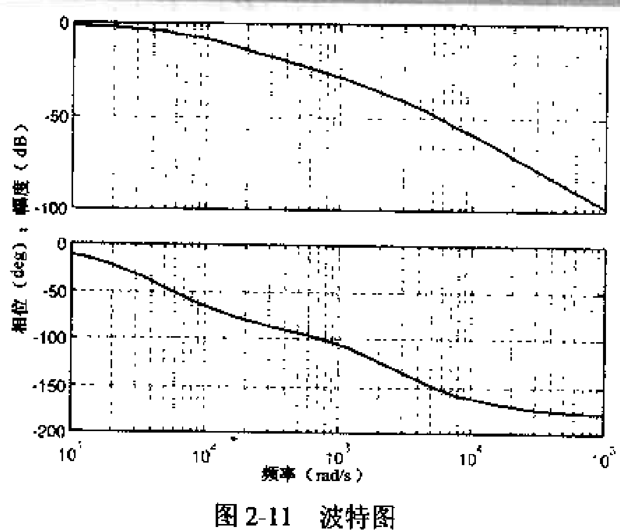


图 2-11　波特图

波特图清楚地显示了两个转折频率，第一个频率主要与机械时间常数相关，第二个频率与电时间常数相关。

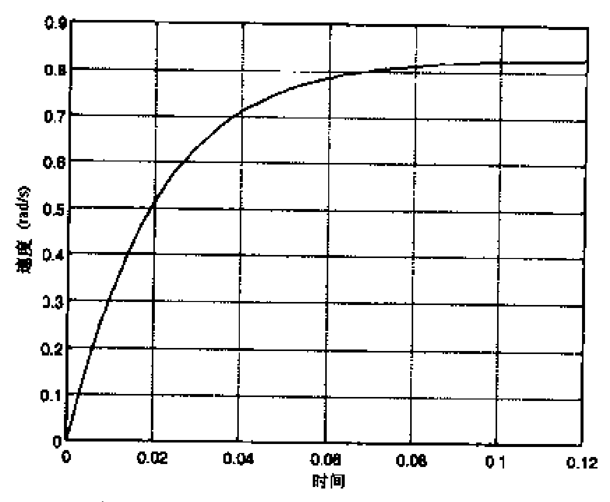


图 2-12　直流电机的阶跃响应曲线

我们要控制电机速度，使位置误差为零。由于这个过程中不包括积分环节，所以控制方法中除了状态反馈外，还需要再串联一个积分器。

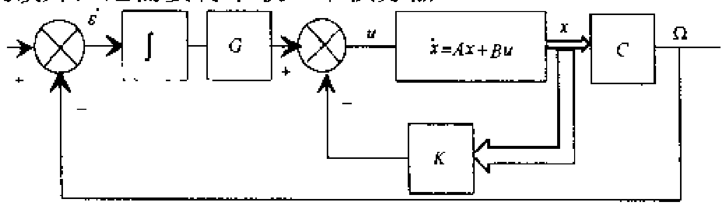


图 2-13　闭环系统结构框图

用下面的等式描述闭环系统：

$$\dot{x} = \boldsymbol{A}x + \boldsymbol{B}u$$
$$y = \boldsymbol{\Omega} = \boldsymbol{C}x$$
$$u = G\varepsilon - Kx$$
$$\dot{\varepsilon} = \boldsymbol{\Omega}_c - \boldsymbol{\Omega} = \boldsymbol{\Omega}_c - \boldsymbol{C}x$$

相应的状态空间描述为：

$$\begin{bmatrix} \dot{x} \\ \dot{\varepsilon} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & 0 \\ -\boldsymbol{C} & 0 \end{bmatrix} \times \begin{bmatrix} x \\ \varepsilon \end{bmatrix} + \begin{bmatrix} \boldsymbol{B} \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \boldsymbol{\Omega}_c$$

把 $u=G\varepsilon-Kx$ 代入上式得：

$$\begin{bmatrix} \dot{x} \\ \dot{\varepsilon} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}-\boldsymbol{BK} & \boldsymbol{BG} \\ -\boldsymbol{C} & 0 \end{bmatrix} \times \begin{bmatrix} x \\ \varepsilon \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \boldsymbol{\Omega}_c$$

◆ 状态反馈向量 K 的表示

计算 $t\to\infty$ 时上面状态方程的极限：

$$\begin{bmatrix} \dot{x}(\infty) \\ \dot{\varepsilon}(\infty) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & 0 \\ -\boldsymbol{C} & 0 \end{bmatrix} \times \begin{bmatrix} x(\infty) \\ \varepsilon(\infty) \end{bmatrix} + \begin{bmatrix} \boldsymbol{B} \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \boldsymbol{\Omega}_c(\infty)$$

考虑到阶跃响应

$$\boldsymbol{\Omega}_c(t) = \boldsymbol{\Omega}_c(\infty) = 1$$

那么

$$\begin{bmatrix} \dot{x}(t)-\dot{x}(\infty) \\ \dot{\varepsilon}(t)-\dot{\varepsilon}(\infty) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & 0 \\ -\boldsymbol{C} & 0 \end{bmatrix} \times \begin{bmatrix} x(t)-x(\infty) \\ \varepsilon(t)-\varepsilon(\infty) \end{bmatrix} + \begin{bmatrix} \boldsymbol{B} \\ 0 \end{bmatrix} [u(t)-u(\infty)]$$

即

$$x_e = x(t) - x(\infty)$$
$$\varepsilon_e = \varepsilon(t) - \varepsilon(\infty)$$
$$u_e(t) = u(t) - u(\infty)$$
$$\begin{bmatrix} \dot{x} \\ \dot{\varepsilon} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & 0 \\ -C & 0 \end{bmatrix} \times \begin{bmatrix} x_e \\ \varepsilon_e \end{bmatrix} + \begin{bmatrix} \boldsymbol{B} \\ 0 \end{bmatrix} u$$

由 $u_e=G\varepsilon_e-Kx_e$，有

$$e = \begin{bmatrix} x_e \\ \varepsilon_e \end{bmatrix}$$

$$\dot{e} = \begin{bmatrix} \boldsymbol{A} & 0 \\ -\boldsymbol{C} & 0 \end{bmatrix} \times e + \begin{bmatrix} \boldsymbol{B} \\ 0 \end{bmatrix} u_e = \hat{\boldsymbol{A}}\mathrm{e} + \hat{\boldsymbol{B}} u_e$$

其中

$$u_e=-\hat{K}e \qquad \hat{K}=[K \quad -G]$$

在上式中应用 Ackermann 公式计算向量 K 和增益 G。

◆ 闭环状态空间描述计算

```
% State representation in closed-loop
Ae=[A zeros(2,1);-C 0],Be=[B ; 0],Ce=[1 0],De=0
Ae =
  1. 0e+005 *
        0         0. 0000   0
        -1.4460 -0. 0300    0
        0. 0000  0          0
Be =
  1. 0e+005 *
      0
1. 2000
      0
Ce =
     1    0
De =
    0
% Comb controllability Matrix
comb=ctrb(Ae,Be)

comb=
  1.0e+012 *
  0          0. 0000      -0. 0004
  0.0000    -0. 0004       1.0628
  0          0             0.0000
% Controllability matrix rank
rang=rank(comb)

rang=
    3
```

显然，系统可控。

配置极点使系统过程具有闭环动态性能：一对复极点和一个单极点，从而绘制理想的指数响应曲线（见图 2-14）。

```
% poles choice of the closed-loop system
wn=2*pi/5e-3;
z=sqrt(2)/2;
p1=-z*wn+(wn*sqrt(1-z^2))*i;
p2=-z*wn-(wn*sqrt(1-z^2))*i;
p3=-1000;
sysc=zpk([],[p1 p2 p3],abs(p1*p2*p3));
[yc,tc]=step(sysc);
figure(2);
plot(tc,yc),grid;
title('Searched for indicial response');
xlabel('Time');
ylabel('Speed in rad/s');
```

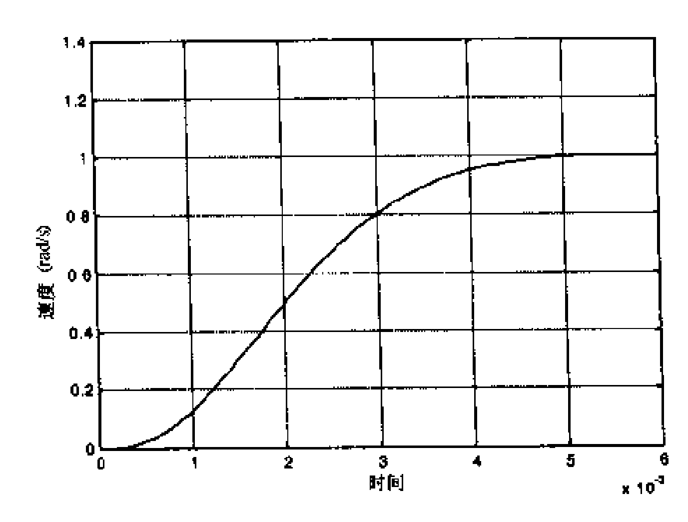


图 2-14　理想指数响应曲线

理想响应的过程响应时间是过程响应时间的 1/10，并且消除了稳态误差。

应用 Ackermann 公式计算反馈向量。

```
% State return vector calculation
% Ackermann formula utilization
p=poly([p1;p2;p3]);
phi=polyvalm(p,Ae);
K1=[0 0 1]*(inv(comb))*phi

K1=
  1.0e+004 *
0.00267640823286  -0.00000018587235  -1.31594725347858
```

分别输入反馈向量 K 和增益 G，计算闭环状态空间描述。

```
% Closed-loop state representation
G=-K1(1,3);
K=[K1(1,1) K1(1,2)];
Ab=[A-B*K B*G; -C 0];
Bb=[0;0;1];
Cb=[C 0];
Db=0;

Ab =
  1.0e+009 *
   0.00000000100000   0                    0
  -0.00335628987944  -0.00000277715318     1.57913670417430
  -0.00000000000000   0                    0
Bb =
    0
    0
    1
Cb =
     1   0   0
Db =
      0
```

由状态空间描述，计算闭环系统的指数响应曲线（见图 2-15）。

```
% Closed-loop system indicial response
figure(3);
[y,x,t]=step(Ab,Bb,Cb,Db);
plot(t,y),grid;
title('Closed-loop process indicial response');
xlabel('Time');
ylabel('Speed in rad/s');
```

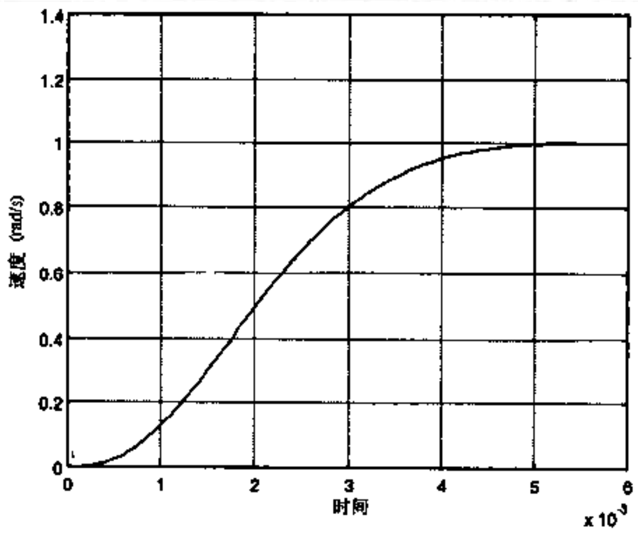


图 2-15　闭环系统的指数响应曲线

计算状态变量的指数响应得：

$$x_1(t) = \Omega(t)$$

$$x_2(t) = \dot{\Omega}(t)$$

$$x_3(t) = \dot{\varepsilon}(t)$$

```
% State variables indicial response
figure(4);
plot(t,x(:,1)),grid;
title(' x1(t)=y(t)Indicial response');
xlabel('Time');
ylabel('x1(t)');
figure(5);
plot(t,x(:,2)),grid;
title('x2(t)=x1''(t)indicial response ');
xlabel('Time');
ylabel('x2(t)');
figure(6);
plot(t,x(:,3)),grid;
title('x3(t)indicial response: error');
xlabel('Time');ylabel('x3(t)');
```

下面的坐标图 2-16 分别对应于状态变量的 3 个阶跃响应：位置、速度和电机轴加速度。

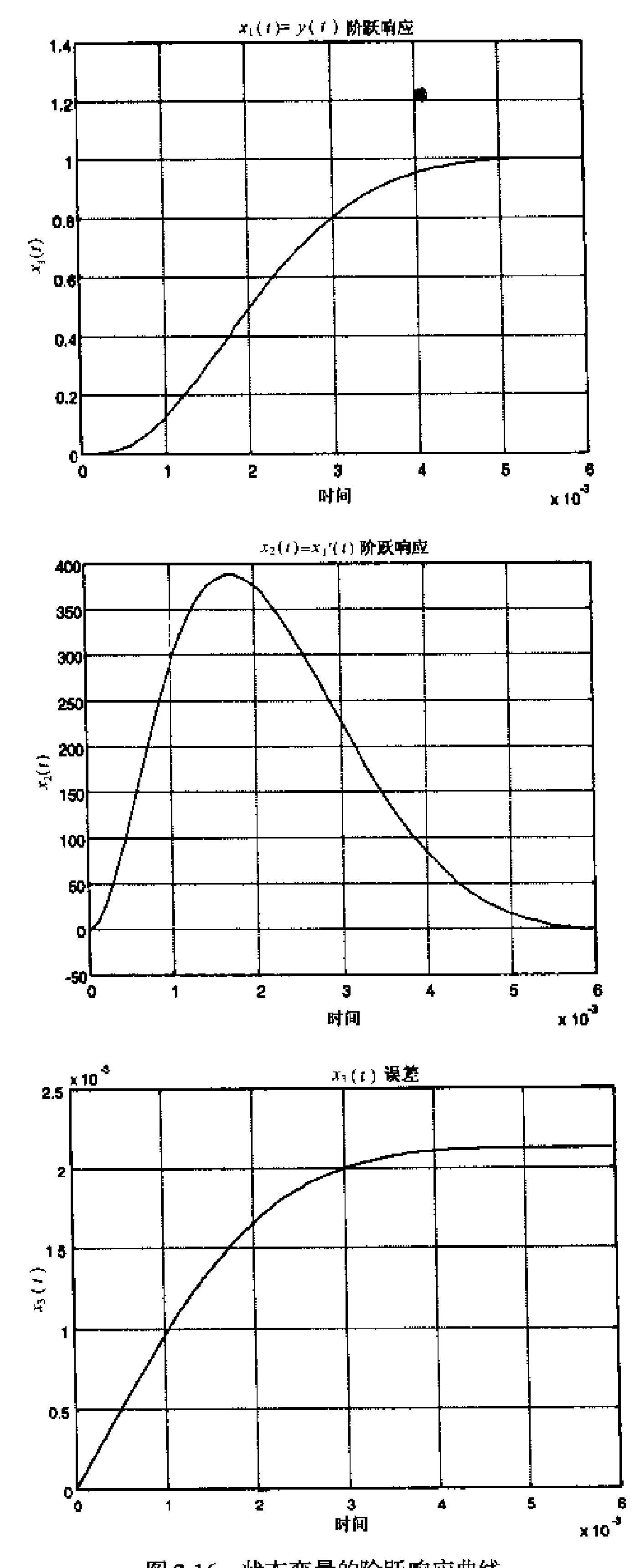


图 2-16　状态变量的阶跃响应曲线

```
% u(t) control evolution
l=length(x);
u=zeros(1,l);
for k=1:l,
   u(k)=-K1*x(k,:)';
end;
figure(7);
plot(t,u),grid;
title('u(t) control voltage ');
xlabel('Time');
ylabel('u(t)');
```

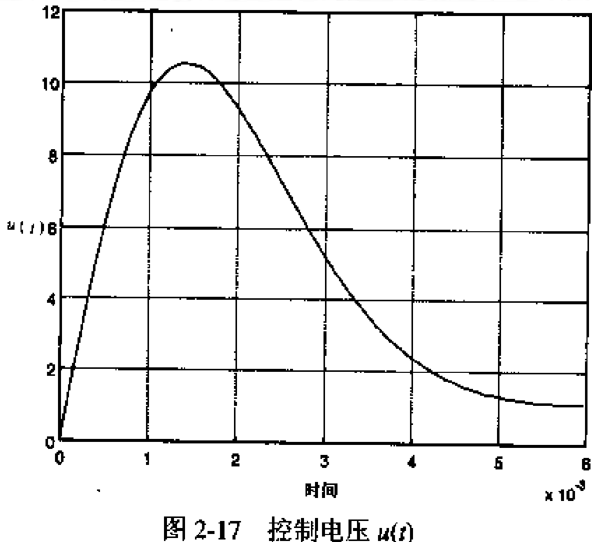


图 2-17　控制电压 $u(t)$

2.6.3　离散系统的极点配置

本节同样以直流电机位置伺服控制为例，不同的是本节中的时间是离散的。例子文件为 state_var3.m。用 c2d 函数求得类似的离散状态模型，采样周期 T_s=0.02 s。

state_var3.m file

```
% Position discrete control system  of the direct current motor
% direct current motor characteristics
r=3; L=0.001;
f=0.002; J=0.01;
k=1.2;
K=k/(k^2+r*f);
T=r*J/(k^2^+r*f);

% A,B,C,D analog model
A=[0 1; 0 -1/T];
B=[0 K/T]';
C=[1 0];
D=0;
sys_ss=ss(A,B,C,D);

% State model discretisation
Te=0.02;
[Ad,Bd]=c2d(A,B,Te)
```

```
Ad =
1.0000   0.0200
     0   0.9960
Bd =
0.0000
0.0035

% Comb controllability matrix
comb=ctrb(Ad,Bd)
% controllability matrix rank
rang=rank(comb)

rang=
    2
```

显然，系统可控。

配置极点使这个伺服控制系统具有一个二阶衰减振荡系统的动态特性。由下列方程可求得极点：

$$p(z)=z^2+p_1z+p_2$$

$$p_1=-2\mathrm{e}^{m\omega_n T_s}\cos\left(\omega_n T_s\sqrt{1-m^2}\right)$$

$$p_2=\mathrm{e}^{-2m\omega_n T_s}$$

```
% Servo control system poles choice
m=sqrt(2)/2;
wn=10*pi;
p1=-2*exp(-m*wn*Te)*cos(wn*Te*sqrt(1-m^2));
p2=exp(-2*m*wn*Te);
p=[1 p1 p2];
sysc=tf(sum(p),p,Te);
figure(2);
step(sysc),grid;
title('Searched for indicial response');
ylabel('Position in rad');
```

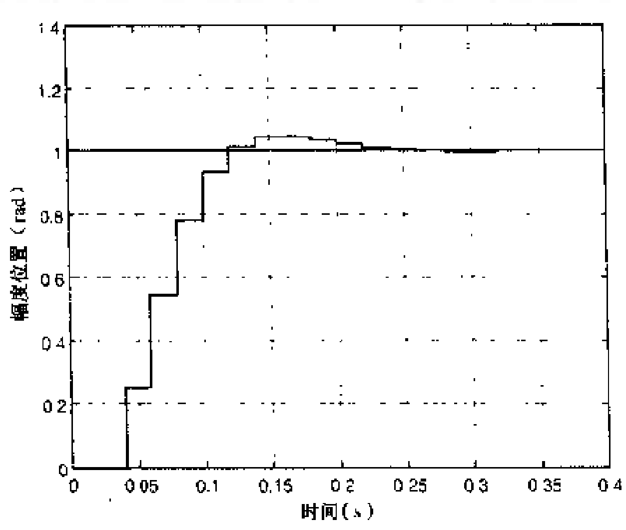


图 2-18　阶跃响应曲线

计算状态反馈向量。

```
 % State return vector calculation
% Ackermann formula utilization
phi=polyvalm(p,Ad);
K=[0 1]*(inv(comb))*phi
K=
  1.0e+003 *
0.2157

% State representation in closed-loop
Abd=Ad-Bd*K;
Bbd=Bd*K(1,1);
Cbd=C;
Dbd=D;

% Closed-loop system indicial response
figure(3);
sysb_ss=ss(Abd,Bbd,Cbd,Dbd,Te);
[y,t,x]=step(sysb_ss);
stem(t,y),grid;
title('Closed-loop system indicial response');
xlabel('Time');ylabel('Position in rad');
```

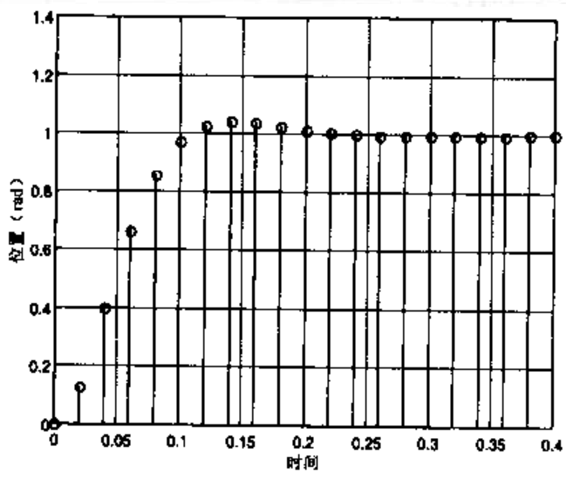


图 2-19　闭环系统阶跃响应曲线

所得的结果与理想动态性能一致。

```
% State variables indicial responses
figure(4);
stem(t,x(:,1)),grid;
title('x1(t)=y(t)indicial response');
xlabel('Time');
ylabel('x1(t)');
figure(5);
stem(t,x(:,2)),grid;
title('x2(t)=x1''(t)indicial response');
xlabel('Time');ylabel('x2(t)');
```

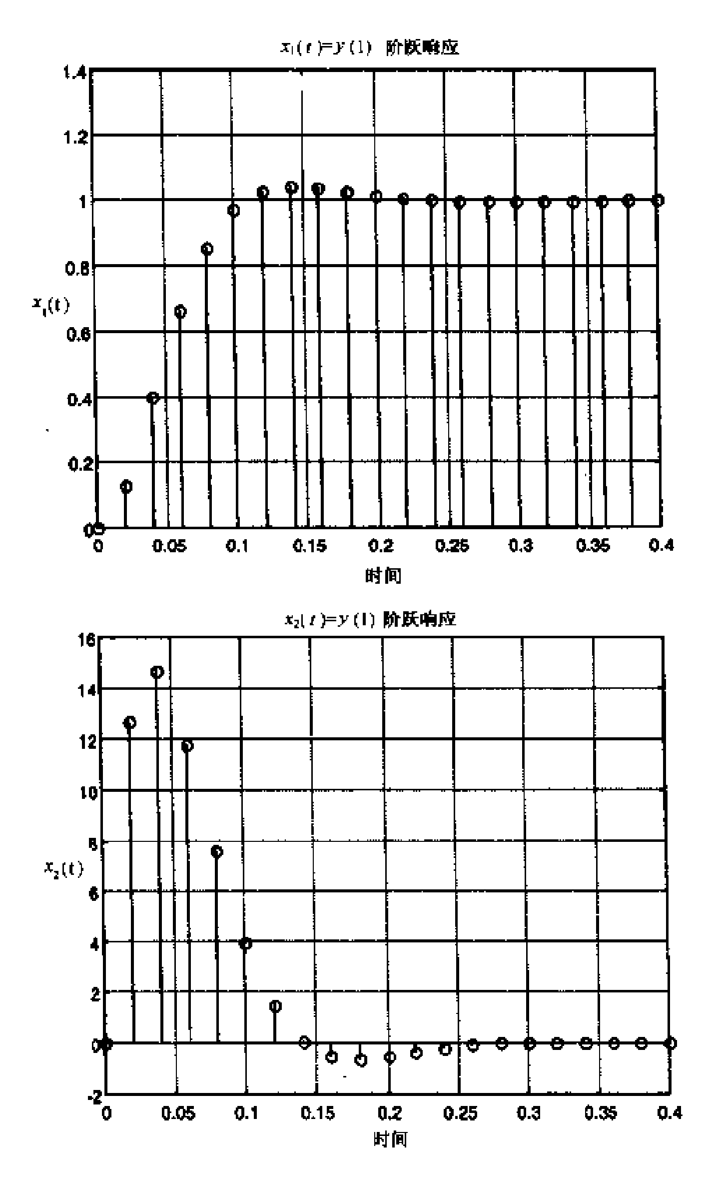


图 2-20　状态变量阶跃响应曲线

2.7　卡尔曼滤波器

卡尔曼滤波器给出了一个应用状态变量概念的公式。而且，不同于其他的递归滤波器结构，它只需要记住前一步的估计结果。考虑过程噪声和测量噪声两个随机变量的状态模型称为随机状态模型。用下面两个方程描述离散状态模型

- 过程方程

$$x(k+1)=\boldsymbol{A}x(k)+\boldsymbol{B}\boldsymbol{u}(k)+w(k)$$

其中 $w(k)$是由于过程模型的不确定性而产生的模型噪声，它可能是最难量化的参数。假定它为均值为零的高斯白噪声：

$$E[w(k)]=0$$

$$E[w(k)w^{\mathrm{T}}(k+n)]=\boldsymbol{Q}$$

$\boldsymbol{Q}$ 是 $w(k)$的 var 矩阵。如果系统不变，那么：

$$Q = \sigma^2_w = \text{cte}$$

- 输出方程

$$y(k) = Cx(k) + v(k)$$

其中 $v(k)$ 是测量噪声，同样假定其为均值为零、不同时刻不相关的高斯白噪声，即

$$E[v(k)] = 0 \qquad E[v(k)v^{\mathrm{T}}(k+n)] = R$$

R 是 v(k)的方差矩阵。如果系统不变，那么

$$R = \sigma_v^2 = \text{cte}$$

◆ 过程状态估计

状态估计分为两步，如图 2-21 所示。

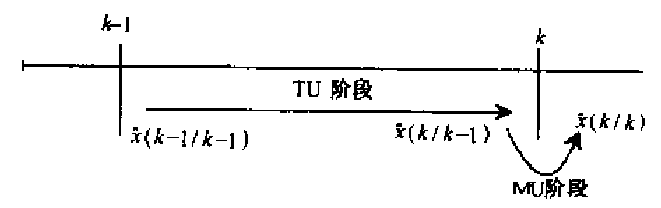


图 2-21 过程状态估计

一个是两个采样周期之间的状态转移阶段，这个阶段叫做 TU(Time Update)阶段：

$$\hat{x}(k/k-1) = A\hat{x}(k-1) + Bu(k-1)$$

在这个阶段，即从$(k-1)T_s$时刻到 kT_s 时刻，可以观察到估计误差方差 $P(k/k-1)$的增加：

$$P(k/k-1) = E\{[x(k) - \hat{x}(k/k-1)][x(k) - \hat{x}(k/k-1)]^{\mathrm{T}}\}$$

$$P(k/k-1) = AP(k-1/k-1)A^{\mathrm{T}} + Q$$

另一个是获得 $y(k)$的 t 时刻过程状态更新阶段，这个阶段叫做 MU（Measurement Update）阶段。这一步由下面的方程描述：

$$\hat{x}(k/k) = \hat{x}(k/k-1) + L(k)[y(k) - C\hat{x}(k/k-1)]$$

其中 $L(k)$为滤波器的自适应增益矩阵。注意，这时由于引入了输出 $y(k)$，减小了误差的方差：

$$P(k/k) = P(k/k-1) + P(k/k-1)C^{\mathrm{T}}[R + CP(k/k-1)C^{\mathrm{T}}]^{-1}CP(k/k-1)$$

增益由下式推导：

$$L(k) = P(k/k-1)C^{\mathrm{T}}[R + CP(k/k-1)C^{\mathrm{T}}]^{-1}$$

◆ 噪声过程，卡尔曼估计滤波器的应用

仍以直流电机过程为例，输出为速度，在这个例子中考虑过程噪声。

假定噪声为均值为零的高斯白噪声。表示模型不确定性的模型噪声为 $w(t)$，测量噪声为 $v(t)$。

卡尔曼估计滤波器可以根据控制信号 $u(k)$和测量输出 $y_b(k)$估计过程输出 $y(k)$和状态变量 $x(k)$。需要的先验知识包括噪声 $w(k)$、$v(k)$的方差，以及如果不为零时它们的互相关性。

考虑到电机的时间常数，以 T_e=0.02s 离散化，对应文件 state_var4.m 如下。

state_var4.m file

```
% Kalman estimator filter
% Direct current motor characteristics
r=3; L=0.001;
```

```
f=0.002; J=0.01;k=1.2;
a0=(k^2+r*f)/(L*J);a1=f/J+r/L;b0=k/(L*J);

% State model
A=[0 1; -a0 -a1];
B=[0 b0]';
C=[1 0];D=0;

% State model discretization
Te=0.02;[Ad,Bd]=c2d(A,B,Te),sys=ss(Ad,Bd,C,D,Te);
```

◆ 过程描述（见图 2-22）

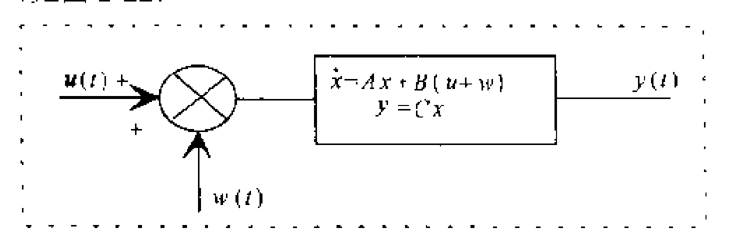


图 2-22　过程结构框图

$$\dot{x} = \boldsymbol{A}x + b(u + w)$$

也可写成如下形式：

$$\dot{x} = \boldsymbol{A}x + [\boldsymbol{B} \quad \boldsymbol{B}][u \quad w]$$

这个式子是有两个输入 u(t)和 w(t)的状态空间描述，可用 ss 函数来计算。

```
% Process state representation
process=ss(Ad,[Bd Bd],C,0,Te,'inputname',{'u' 'w'},...
   'outputname',{'y'});
a =
                    X1              X2
         x1       0.38167         0.00013
         x2     -18.70070        -0.00634
b =
                     u               w
         x1       0.51314         0.51314
         x2      15.51925        15.51925
c =
                    x1              x2
          y       1.00000            0
d =
                     u               w
          y          0               0

Sampling time  :  0.02
Discrete -time   system
```

$w(t)$和 $v(t)$的方差分别为 Q 和 R。N 为 $w(t)$和 $v(t)$的互相关矩阵。卡尔曼函数根据 Q、R、N 和过程状态描述计算卡尔曼估计滤波器。在这个例子中，模型噪声和测量噪声不相关，即 N=0。

```
% noises
n=100;
```

```
w=0.1*randn(n,1);
v=0.3*randn(n,1);

Q=std(w).^2;
R=std(v).^2;
wv=cov(v,w);
N=wv(1,2);

% Kalman estimator
[F_Kalman,L,P,M]=kalman(process,Q,R);
```

◆ 模型描述+测量噪声+卡尔曼滤波器（见图 2-23）

测量噪声加上模型输出组成输出信息。

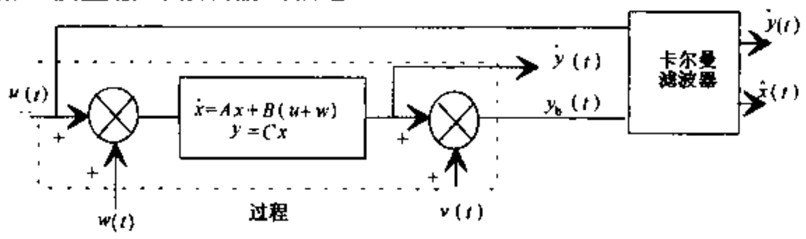


图 2-23　系统结构框图

- 过程输入向量

$$[u \quad w \quad v]^{\mathrm{T}}$$

- 过程输出向量

$$[y \quad y_b]^{\mathrm{T}}$$

过程可由如下系统表示：

$$\dot{x}=\boldsymbol{A}x+\begin{bmatrix}\boldsymbol{B} & \boldsymbol{B} & \begin{matrix}0\\0\end{matrix}\end{bmatrix}[u \quad w \quad v]^{\mathrm{T}}$$

$$[y \quad y_b]^{\mathrm{T}}=\begin{bmatrix}\boldsymbol{C}\\\boldsymbol{C}\end{bmatrix}x+\begin{bmatrix}0 & 0 & 0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}u\\w\\v\end{bmatrix}$$

建立包含测量噪声的一般模型的状态空间描述。

```
% State representation process + measurement noise
a=Ad;
b=[Bd Bd [0;0]];
c=[C;C];
d=[0 0 0;0 0 1];
process=ss(a,b,c,d,Te,'inputname',{'u' 'w' 'v'},...
   'outputname',{'y' 'yb'})
```

然后把这个过程和卡尔曼滤波器联系起来，得到整个过程的状态空间描述，其输入输出如下：

- 输入向量

$$[u \quad w \quad v]^{\mathrm{T}}$$

- 输出向量

$$\begin{bmatrix} y & y_b & \hat{y} & x_1 & x_2 \end{bmatrix}^T$$

函数 **parallel** 用来把卡尔曼滤波器和过程联系起来。然后，函数 **feedback** 用来创建把过程输出 y_b 当做卡尔曼滤波器输入的反馈环节，最后由状态模型得到输出。

```
% Parallel placing process + Kalman filter
sysp=parallel(process,F_Kalman,1,1,[],[]);

% yv feedback
syspb=feedback(sysp,1,4,2,1);

% yv input cancellation
syspb=syspb([1 2 3 4 5],[1 2 3]);

syspb

a =
                 ?           ?        x1_e        x2_e
     ?     0.38167     0.00013           0           0
     ?   -18.70070    -0.00634           0           0
  x1_e     0.03278           0     0.34889     0.00013
  x2_e    -1.60625           0   -17.09445    -0.00634
```

```
b =
                      w           v           u
        ?       0.51314           0     0.51314
        ?      15.51925           0    15.51925
     x1_e             0     0.03278     0.51314
     x2_e             0    -1.60625    15.51925
c =
                 ?         ?      x1_ e        x2_e
  y        1.00000         0          0           0
  yb       1.00000         0          0           0
  y_e      0.08533         0    0.91467           0
  x1_e     0.08533         0    0.91467           0
  x2_e     1.66767         0   -1.66767     1.00000
d =
                      w           v           u
            y         0           0           0
            yb        0     1.00000           0
            y_e       0     0.08533           0
            x1_e      0     0.08533           0
            x2_e      0     1.66767           0

Saampling time : 0.02
Discrete- time system.

syspb.input

'w'
'v'
```

```
'u'
>> syspb.output

'y'
'yb'
'y_e'
'x1_e'
'x2_e'
```

以一个平方信号作为过程输入，利用 **lsim** 函数，分别计算无噪声过程响应和有噪声过程响应来加以比较。

```
% Control square signal
t=[0:Te:(n-1)*Te];
u=square(2*pi*t)';

% Inputs representation
figure(1);
plot(t,u);
title('Control signal');
xlabel('Time'),ylabel('u(t)');
axis([0 2 -1.2 1.2]);

figure(2);
plot(t,w);
title('Process noise');
xlabel('Time'),ylabel('w(t)');
figure(3);
plot(t,v);
title('Measurement noise');
xlabel('Time'),ylabel('v(t)');

% Noisefree process simulation
[yp,t,xp]=lsim(sys,u,t);

% Full model simulation
[outputs,x]=lsim(syspb,[w v u],t);

% Results representation
figure(4);
plot(t,outputs(:,1)),hold on;
plot(t,yp,'r:'),hold off;
title('Output process with noise');
xlabel('Time'),ylabel('y(t)');

figure(5);
plot(t,outputs(:,2)),hold on;
plot(t,yp,'r:'),hold off;
title('process with noise + measurement noise outputs');
xlabel('Time'),ylabel('yb(t)');

figure(6);
```

```
plot(t,outputs(:,3)),hold on;
plot(t,yp,'r:'),hold off;
title('Estimated process output');
xlabel('Time'),ylabel('ye(t)');

figure(7);
plot(t,outputs(:,4)),hold on;
plot(t,yp,'r:'),hold off;
title('x1(t) estimation');
xlabel('Time'),ylabel('x1(t)');

figure(8);
plot(t,outputs(:,5)),hold on;
plot(t,xp(:,2),'r:'),hold off;
title('x2(t) estimation');
xlabel('Time'),ylabel('x2(t)');
```

下列图 2-24、图 2-25 和图 2-26 分别为控制信号、模型噪声和测量噪声波形图。

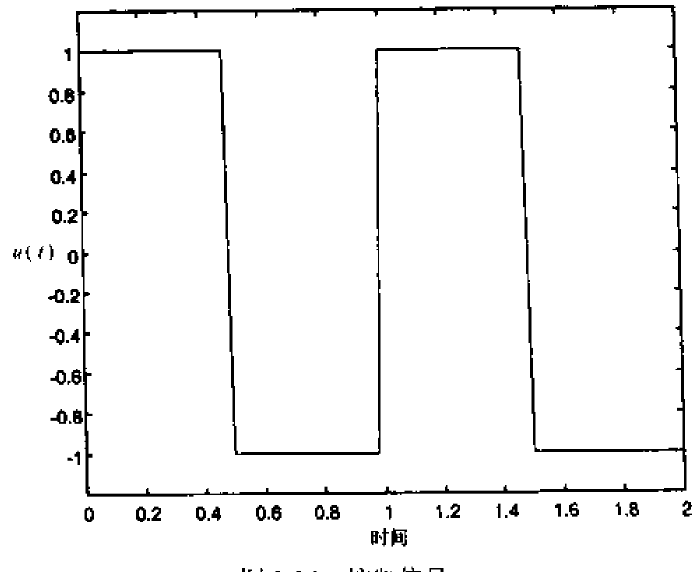


图 2-24 控制信号

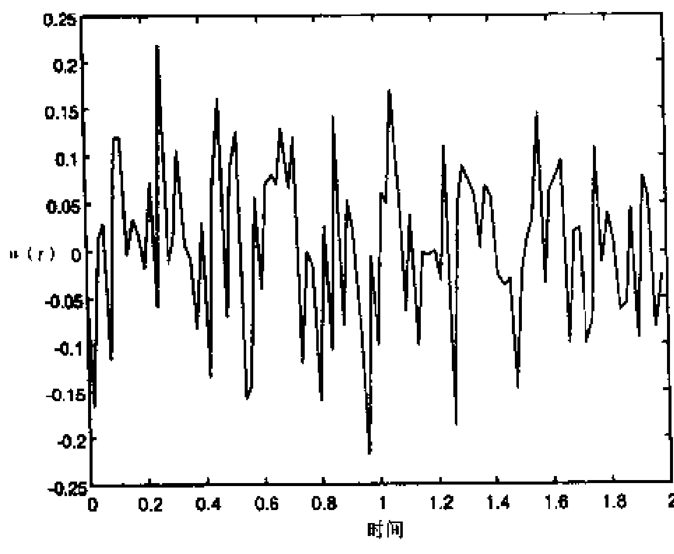


图 2-25 模型噪声

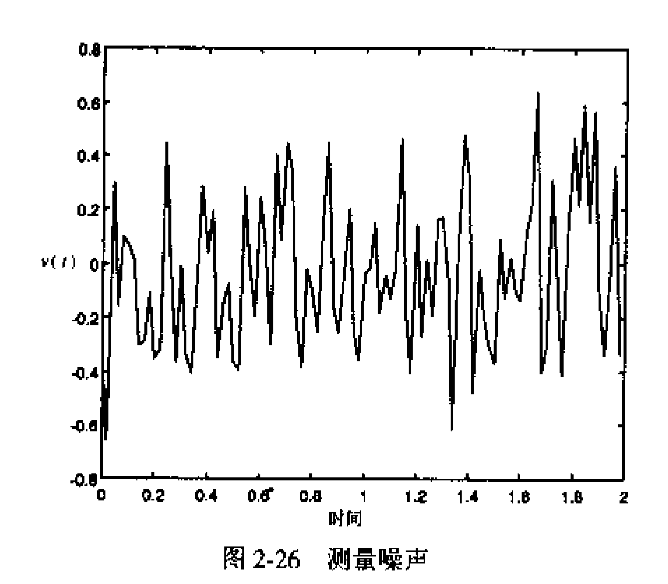


图 2-26　测量噪声

下面分别为有噪声过程（见图 2-27）和无噪声过程的响应曲线。

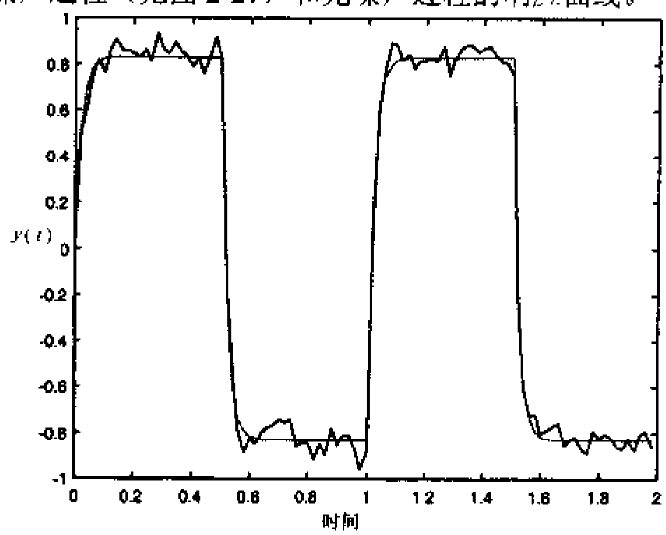


图 2-27　有噪声过程的响应曲线

如果考虑测量噪声（变量 0.3），得到图 2-28 所示的响应。

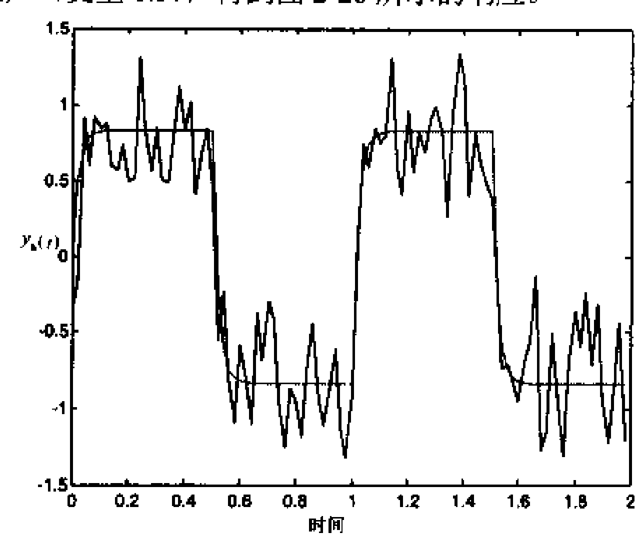


图 2-28　无噪声过程的响应曲线

卡尔曼滤波器的估计输出接近于无噪声过程的输出（没有测量和模型噪声）。

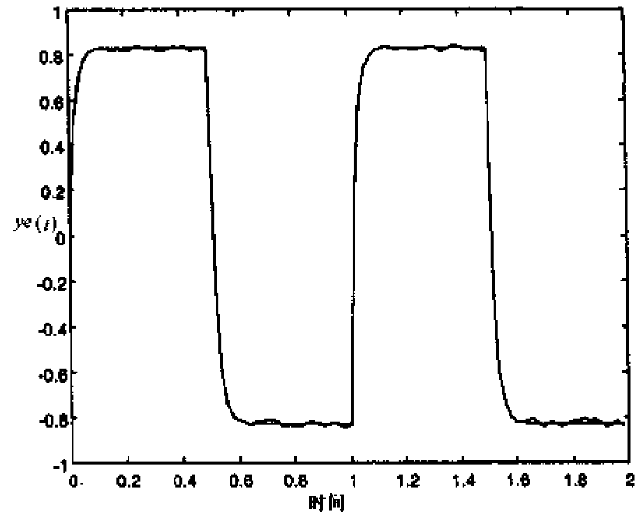


图 2-29　估计过程输出

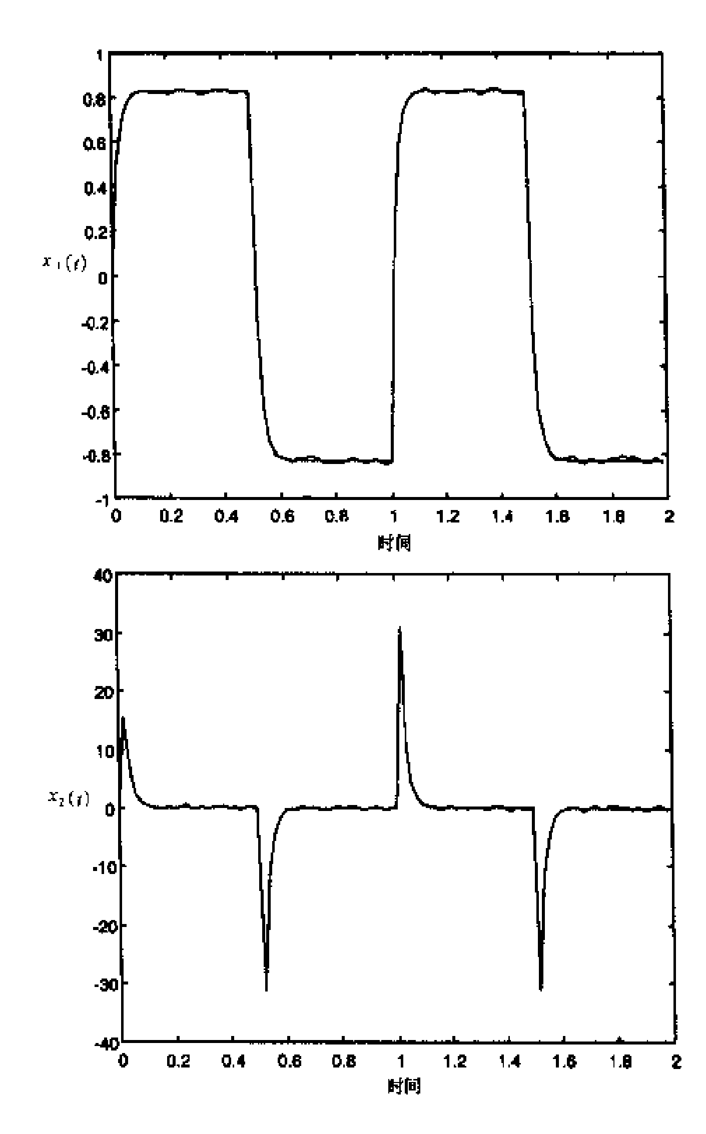


图 2-30　状态估计

状态估计也令人满意。所以，这样的过程避免了使用昂贵的传感器或者可以在测量比较困难或有噪声污染时使用。

计算各个误差方差。

```
% Errors variance calculation
% Modeling noise
e1=yp-outputs(:,1);
ec1=std(e1)

% Modeling and measurement noises
e2=yp-outputs(:,2);
ec2=std(e2)

% Filtered process
e3=yp-outputs(:,3);
ec3=std(e3)

ec1 =
   0.0637
ec2 =
0.3204
ec3 =
   0. 0136
```

估计输出信号的误差方差约为测量输出信号误差方差的 1/23，约为无测量噪声时输出信号误差方差的 1/5。

2.8 随机离散卡尔曼预测器

卡尔曼预测器是用$(k-1)T_s$时刻的输出 $y(k-1)$预测 kT_s 时刻的过程状态 $x(k/k-1)$。这时两个阶段的顺序相反。首先是 MU（Measurement Update）阶段，然后是 TU（Time Update）阶段。

- MU（Measurement Update）阶段

$$\hat{x}(k-1/k-1)=\hat{x}(k-1/k-2)+L(k-1)[y(k-1)-C\hat{x}(k-1/k-2)]$$

- TU（Time Update）阶段

$$\hat{x}(k/k-1)=A\hat{x}(k-1/k-1)+Bu(k-1)$$

把 MU 中得到的 $\hat{x}(k-1/k-1)$表达式代入上式：

$$\hat{x}(k/k-1)=A\hat{x}(k-1/k-2)+AL(k-1)[y(k-1)-C\hat{x}(k-1/k-1)]$$

其中自适应增益为：

$$L(k-1)=P(k-1/k-2)C^{\mathrm{T}}\left[R+CP(k-1/k-2)C^{\mathrm{T}}\right]^{-1}$$

误差方差矩阵为：

$$P(k/k-1)=A[P(k-1/k-2)-L(k-1)CP(k-1/k-2)]A^{\mathrm{T}}+Q$$

其中 Q 和 S 分别为模型误差 $w(k)$方差阵和测量误差 $v(k)$方差阵。

◆ 卡尔曼预测器在有噪声过程的应用

仍以直流电机过程为例，输出为速度，在这个例子中考虑过程噪声。

假定噪声为均值为零的高斯白噪声。表示模型不确定性的模型噪声为 $w(k)$，测量噪声为 $v(k)$。

卡尔曼预测滤波器可以根据已知的前一步状态 $x(k-1)$和测量输出 $y_b(k)$预测状态变量 $x(k)$。需要的先验知识包括噪声 $w(k)$、$v(k)$的方差，以及如果不为零时它们的互相关系数。

以 T_e=0.02s 离散化，对应文件 state_var5.m 如下。

state_var5.m file

```
% Kalman predictor

% Direct current motor characteristics
r=3; L=0.001;
f=0.002; J=0.01;
k=1.2;
a0=(k^2+r*f)/(L*J);
a1=f/J+r/L;
b0=k/(L*J);

% State model
A=[0 1; -a0 -a1];
B=[0 b0]';
C=[1 0];
D=0;

% State model discretization
Te=0.02;
[Ad,Bd]=c2d(A,B,Te)
sys=ss(Ad,Bd,C,D,Te);

% noises
n=100;
w=0.1*randn(n,1);
v=0.3*randn(n,1);
Q=std(w).^2;
R=std(v).^2;

% Control signal
t=0:Te:(n-1)*Te;
u=sin(2*pi*t);
for i=1:n,
   if i<n/4
      u(i)=-1;
   elseif i<n/2
      u(i)=1;
   else
      u(i)=1+sin(4*i*pi/n);
   end;
end;

% State and output with noises evolution
x=zeros(2,n);
for i=2:n,
   x(:,i)=Ad*x(:,i-1)+Bd*u(i-1);
```

```
  xb(:,i)=Ad*x(:,i-1)+Bd*u(i-1)+[w(i-1);w(i-1)];
  yb(i)=C*x(:,i)+v(i);
end;

% State prediction
K=zeros(2,n);
P=eye(2); P11=zeros(1,n);
xe=zeros(2,n);
for i=1:n-1,
  K(:,i)=P*C'/(R+C*P*C');
  P=Ad*(P-K(:,i)*C*P)*Ad'+Q;
  P11(1,i)=P(1,1);
  xe(:,i+1)=Ad*xe(:,i)+Ad*K(:,i)*(yb(i)-C*xe(:,i))+Bd*u(i);
end;

% Signals representation
figure(1);
plot(t,u);
title('Control signal');
xlabel('Time'),ylabel('u(t)');
axis([0 2 -1.2 2.2]);
figure(2);
plot(t,xe(1,:)),hold on;
plot(t,x(1,:),'r:'),hold off;
title('xe1(t) predicted state variable ');
xlabel('Time'),ylabel('xe1(t)');
figure(3);
plot(t,xe(2,:)),hold on;
plot(t,x(2,:),'r:'),hold off;
title('xe2(t) predicted state variable ');
xlabel('Time'),ylabel('xe2(t)');
figure(4);
plot(t(1:20),P11(1,1:20)),grid;
title('Predicted error variance');
xlabel('Time'),ylabel('P11(t)');
```

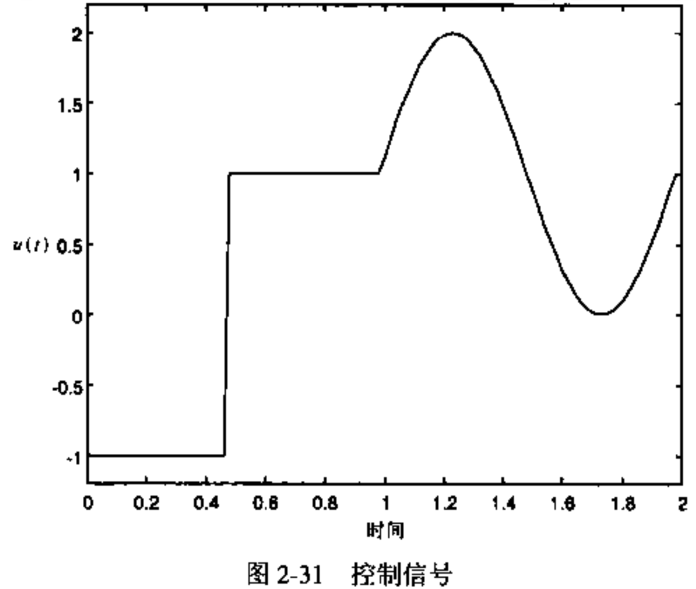


图 2-31　控制信号

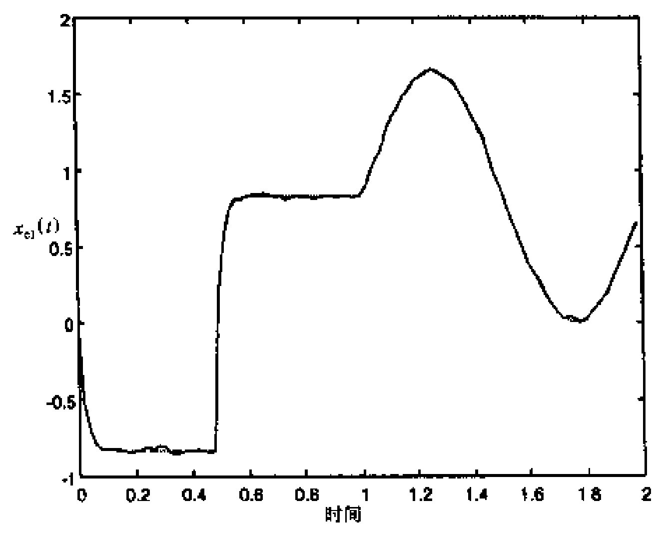


图 2-32　状态变量 $x_{e1}(t)$的预测结果

从图 2-32 中可以看出，代表过程速度的状态变量 $x_{e1}(t)$的预测结果即使控制量变化时也很好，加速度 $x_{e2}(t)$的预测精度稍微差一些，但看上去也十分令人满意（见图 2-33）。

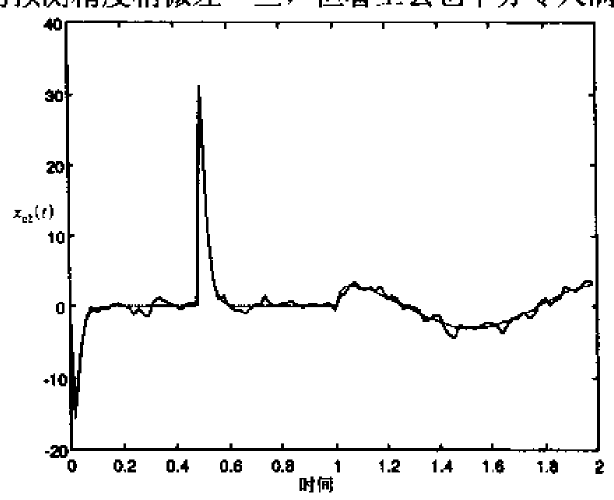


图 2-33　状态变量 $x_{e2}(t)$的预测结果

当控制量为常数时，预测误差方差约在 0.1s 收敛到零。

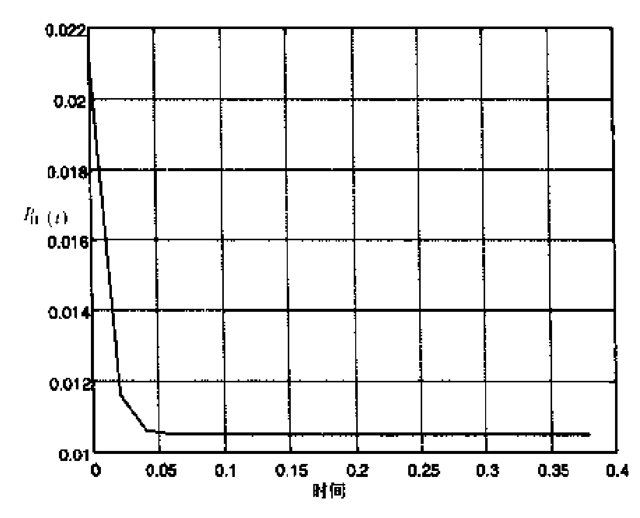


图 2-34　预测误差方差

图 2-35 为卡尔曼滤波器增益向量前 20 次迭代的系数变化。

```
% Predictor coefficients evolution
figure(5)
plot(t(1:20),K(1,1:20)), hold on
plot(t(1:20),K(2,1:20)), hold off, grid
title('Predictor gain vector coefficients')
xlabel('Time'), gtext('K1'), gtext('K2')
```

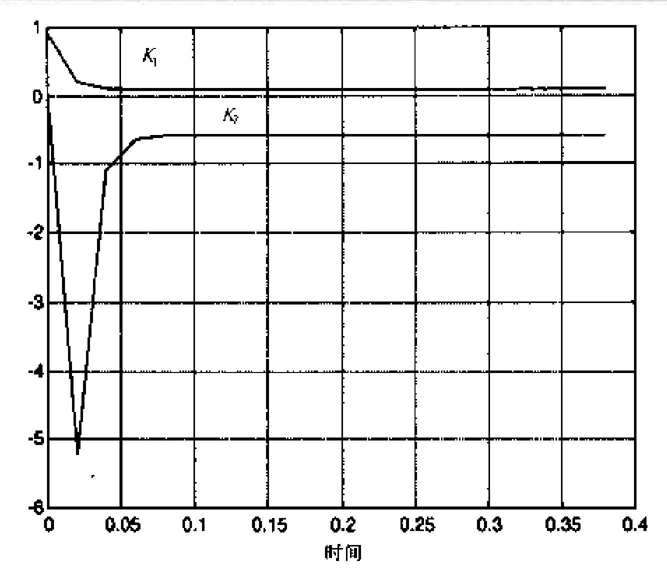


图 2-35　卡尔曼滤波器增益向量前 20 次迭代的系数变化

第 3 章　模糊逻辑控制

3.1　基本原理

模糊集能够定义一个元素隶属于一个集合的程度，即它在多大程度上属于这个集合。用 0 到 1 之间的数来表示隶属度。

举例来说，把人的年龄从 0 到 100 划分为三类：青年、中年、老年。

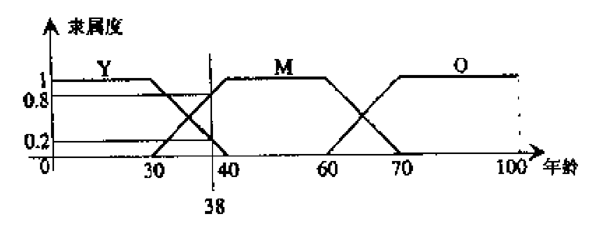


图 3-1　隶属度定义

根据这一逻辑，一个 38 岁的人，20％属于青年，80％属于中年，所以他对于模糊集 Y 的隶属度是 0.2，对于模糊集 M 的隶属度是 0.8。用隶属函数来表示模糊集 Y、M 和 O（Z 对应于 Y，梯形对应 M，S 对应 O）。隶属函数有几种表示法：三角法、高斯法等等。

模糊逻辑应用最广的领域是在工业过程的控制中，这些工业过程或者没有一个数学模型，或者其数学模型有很强的非线性。

以反转摆为例，通过固定在其上的小车施力使其保持在垂直位置，如图 3-2 所示。

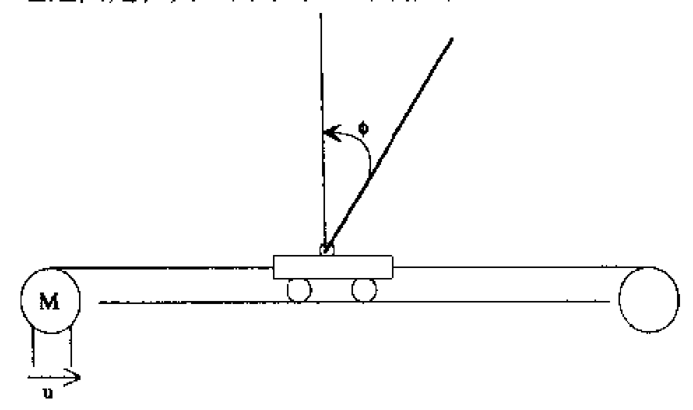


图 3-2　反转摆与小车示意图

模糊调节器的实现需要三个阶段，如图 3-3 所示。

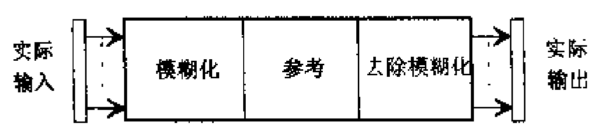


图 3-3　实现模糊调节器的三个阶段

3.2 模糊调节器的实现

3.2.1 模糊化

模糊化包括定义输入输出变量的模糊集。对于每一个变量，必须知道它的取值范围。大多数情况下，模糊调节器的输入都是过程输出与给定信号之间的误差以及误差的变化。由于误差的导数可表示其变化，所以控制律可如图 3-4 所示。

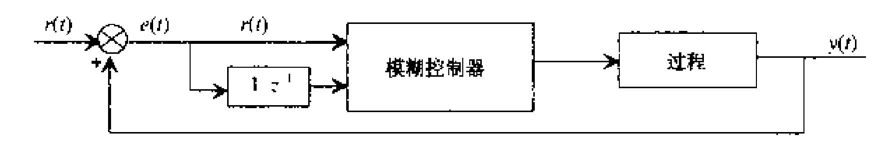


图 3-4 模糊控制

在反转摆的例子中，给定信号 $\phi=0$，所以误差和它的变化分别是：

$$e(t)=y(t)$$

$$\Delta e(t)=y(t)-y(t-1)$$

如果角度值超出了[−20°，20°]的范围，就认为反转摆不可恢复，并且它变化的最大绝对值是 10 %。每一个变量都分别定义在三个模糊集上，这些模糊集用三角隶属函数来定义（N 表示负模糊集，Z 表示零，P 表示正模糊集）。如果小车发动机电压（即调节器输出）限定在[−10V，10V]的范围，将它定义在 5 个三角形的模糊集上（NG：负的绝对值很大，N：负值，Z：零，P：正值，PG：正值很大）。模糊化的过程包括输入输出变量的模糊定义。

误差是输出和给定信号（零角）的差。误差变化是 kT 时刻的误差相对于前一时刻（k−1）T 时刻的变化。设误差和它的变化的最大绝对值分别等于 20 和 10，就得到如图 3-5 所示的模糊集，可用来定义调节器的输入输出变量。

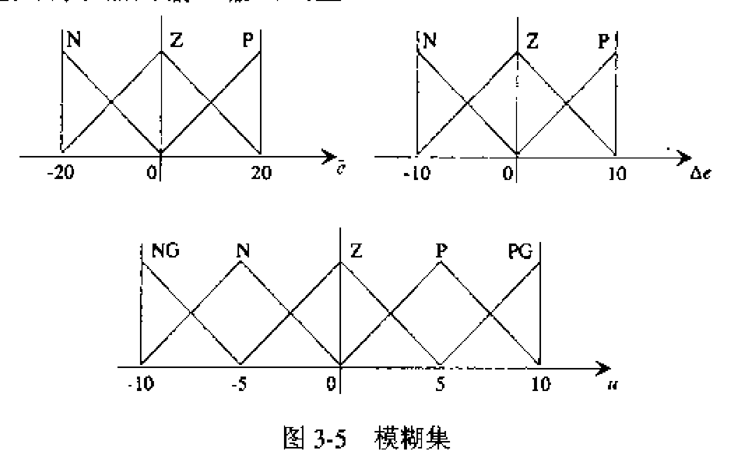


图 3-5 模糊集

3.2.2 推理阶段

这个阶段将建立模糊规则。这些规则能够根据误差和它的变化得出控制信号的值。

通常控制信号是由误差和其变化共同决定的。在反转摆的例子中，必须使其快速返回目标位置。模糊规则将输出变量和输入变量联系起来决定模糊结论或推论。

一个模糊规则包括一个前提，如“如果误差是负值且误差变化也是负值”，和一个模糊推论，如“那么 u 是很大的正值”。在每个输入变量都定义在三个模糊集上时，可得到 9 个模糊规则，如下所示即为反转摆例子的推理表。

表 3-1 反转摆例子的推理表

Δe \ e	N	Z	P
N	PG	P	Z
Z	P	Z	N
P	Z	N	NG

从表中可以看出，模糊调节器每个采样间隔输出的控制量是在前一个采样间隔控制量的基础上增加一个增量。有 3 种情形，输出命令是 Z，这是要保持控制量不变，例如：

- 误差是 Z，它的变化也是 Z（钟摆在目标位置）；
- 误差是 P，它的变化是 N；
- 误差是 N，它的变化是 P。

确定了模糊规则，还需要计算输出变量相对于对应模糊集的隶属度。5 个模糊推论对应于 5 个输出变量模糊集。

1）If (e is N) AND (Δe is N) then u is PG,

2）If (e is N) AND (Δe is Z) OR (e is Z) AND (Δe is N) then u is P,

3）If (e is Z) AND (Δe is Z) OR (e is P) AND (Δe is N) OR (e is N) AND (Δe is P) then u is Z,

4）If { (e is P AND (Δe is Z)) OR {(e is Z) AND (Δe is P) } then u is N,

5）If (e is P) AND (Δe is P) then u is NG.

每个规则都由算子 AND/OR 连接的前提和紧跟在“Then”算子后的结论组成。在 Mamdani 方法中，min 算子对应 AND，max 算子对应 OR。如果每条规则的前提都有不为零的隶属度，几条规则可以同时被激发。这取决于采用的隶属函数。在采用高斯型隶属函数时，所有规则在每一个采样间隔或多或少都会被激发。

规则合成是指用算子 MAX 来确定一个惟一的输出变量，类似于规则之间用 OR 算子相连。下面以规则 1 和 2 的合成来具体说明。

规则 1 的每个前提都有其隶属函数 PG，分别由误差 e 或 Δe 在 N 集上的隶属度决定。由于两个前提由 AND 相连，u 在 PG 集上的隶属度由两个隶属度的最小值决定，如图 3-6 所示。

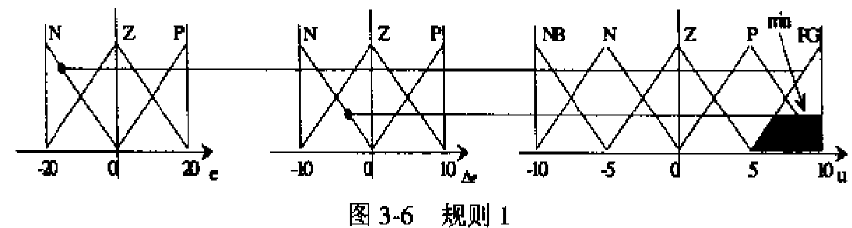


图 3-6 规则 1

规则 2 的两个前提由 OR 相连。每个前提的运算同规则 1。用 max 代替 OR，取两个

隶属度的最大值，如图 3-7 所示。

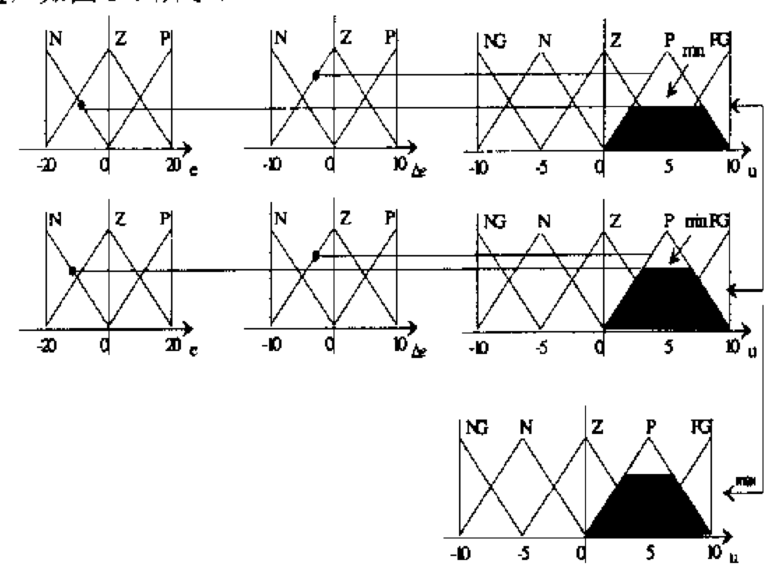


图 3-7　规则 2

规则 1 和规则 2 的合成取上述所得的输出变量在定义集（讨论域）上隶属度的最大值即可，如图 3-8 所示。

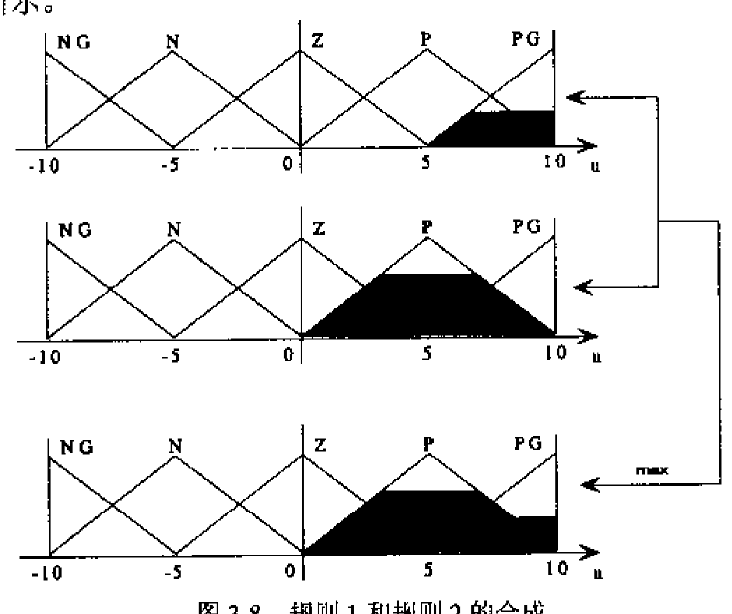


图 3-8　规则 1 和规则 2 的合成

如果规则 1 和规则 2 同时被激发，输出变量的隶属函数即如上图所示。这种方法被称为“max-min 推理”。还有一种“sum-prod”法是由 AND 的结果和 OR 的一半和构成。

3.2.3　去除模糊化

模糊化是计算每个输入变量在其对应模糊集上的隶属度。去除模糊化即进行相反的运算，也就是说，根据推理阶段得到的值计算实际的输出值。有几种去除模糊化的方法，其中用得最多的是 gravity 法。

模糊逻辑工具箱中包括几种去除模糊化的方法，如

- centroid　重心法；

- bisector　表面二分法；
- mom　最大值平均法；
- som　取最大绝对值的最小值法；
- lom　取最大绝对值的最大值法。

defuzzif.m 文件对每一种方法都进行了计算。

defuzzif.m file

```
% observation of different defuzzification methods
% on the only trapezoidal membership function: trapmf
clear all, close all, clc,
result = [];
% interval of definition of the variable
x = -10:0.1:10
% trapezoidal membership function
mf = trapmf(x, [-10, -8, -4, 7]);
% several types of defuzzification
type = char (' lom', ' centroid', ' mom', ' som', ' bisector');
style = char('-',':','-.',':','-');
figure (' name', ' Different methods of defuzzification')
for i = 1:5
  subplot(2,1,(i <3)+2*(i> = 3)), plot(x, mf);
  axis([min(x) max(x) 0 1.2]), hold one, home, type(i,:)
  xx = defuzz(x, mf, wnoblank(type(i,:)))
  plot([xx xx], [0 1.2], wnoblank(style(i,:)));
  plot(xx, 0.0, ' o'), text(xx+0.2, 0.1*(1+i), type(i,:));
  result = [result xx];
  xlabel ('Push on a key to continue! ')
end
subplot(211), xlabel ('')
subplot(212), xlabel (' methods of defuzzification')
% display of vectors of values and types
disp (' type and result of the defuzzification')
for i = 1:length(result)
  disp([type(i,1:length(type)) blanks(5) num2str(result(1,i))])
end
```

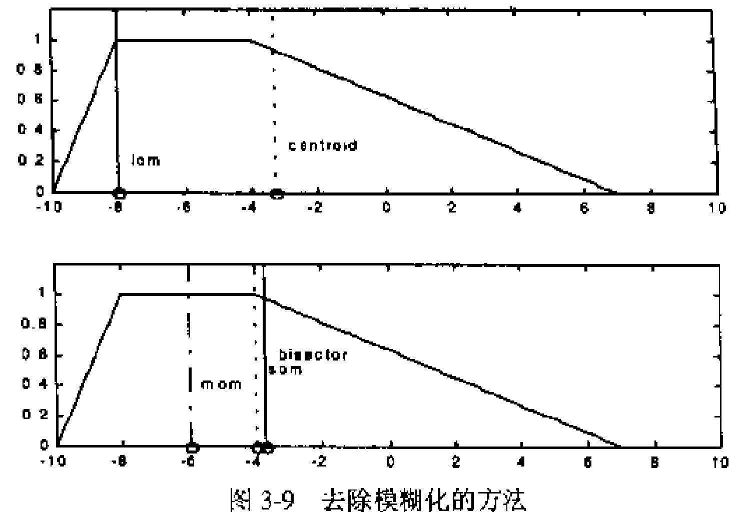


图 3-9　去除模糊化的方法

```
type and result of defuzzification
 lom            -8
 centroid       -3.2857
 mom            -6
 som            -4
 bisector       -3.7
```

在这些方法中，最常用的是重心法 centroid 和 mom 法。重心法计算推理阶段所得表面的重心并把它投影到水平轴上，这样可以得到输出变量以其隶属度作为权值的平均值。

```
>> flop(0),val_centr = sum(x. *mf)./ sum(mf), nb_oper = flops
val_centr = -3.2857
nb_oper = 604
```

这个方法需要 604 浮点数运算。

“mom”法计算最大隶属度（在前面的图例中为 1，在-8~-4 之间，步长为 0.1）的均值较为简单。

```
>> val_com = mean(x(find(mf = = max(mf) ) ) )
>>flops(0)
>> val_com = mean(x(find(mf = = max(mf) ) ) )
>> nb_oper = flops
val_com =  -6.0000
nb_oper =   44
```

在前面的梯形例子中，它等价于下面的简单的算术平均值的计算。

```
》mean ( -8 : 0. 1 :-4)
ans = -6 .0000
```

二分法计算把所得表面等分为 2 份垂直对应的横坐标值。

```
flops(0), surf_tot = sum(mf); surf = 0;
% calculus of the total surface step by step
        for k = 1:lengh(x),
                surf = surf + mf(k);
                if surf>= surf_tot/2
                  break
                end
        end
val_bisector = x(k), nb_oper = flops
val_bisector =  -3.7000
nb_oper =  329
```

Som 法和 lom 法分别取最大值对应的最小值和最大值，在前面的例子分别对应-8 和-4。

```
>>flops(0), val_som = min(x(find(mf = = max(mf) ) ) )
>>nb_oper = flops
val_som =
   -8
```

```
nb_oper =
     0
>>flops(0), val_som = max(x(find(mf = = max(mf) ) ) )
>>nb_oper = flops
val_som =
    -4
nb_oper =
     0
```

后两种方法不需要任何浮点运算，因为它们只关心最大值的范围，即只关心 min 和 max。由于只考虑到了一个量（最小或最大），这个值是很不精确的。所有方法中最精确的是重心法，因为它根据其隶属度大小考虑到了讨论域中所有的值。

图 3-10 表明浮点运算的数量从 mom 法到 centroid 法随着其离散精度的增加而线性上升。

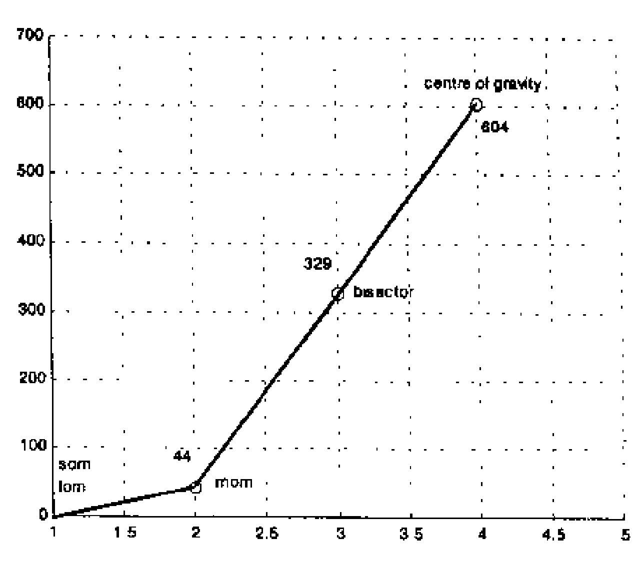


图 3-10　去除模糊化方法的浮点数变化

3.3　模糊逻辑工具箱的图形界面

用 fuzzy 命令打开 FIS 编辑器的图形界面（见图 3-11），在其中可以定义整个模糊系统。默认时，这个界面提供 Mamdani 法的单输入单输出。OR 和 AND 算子分别由 max 和 min 实现，蕴含关系用 min，用 max 合成规则，用重心法（centroid）去除模糊化。

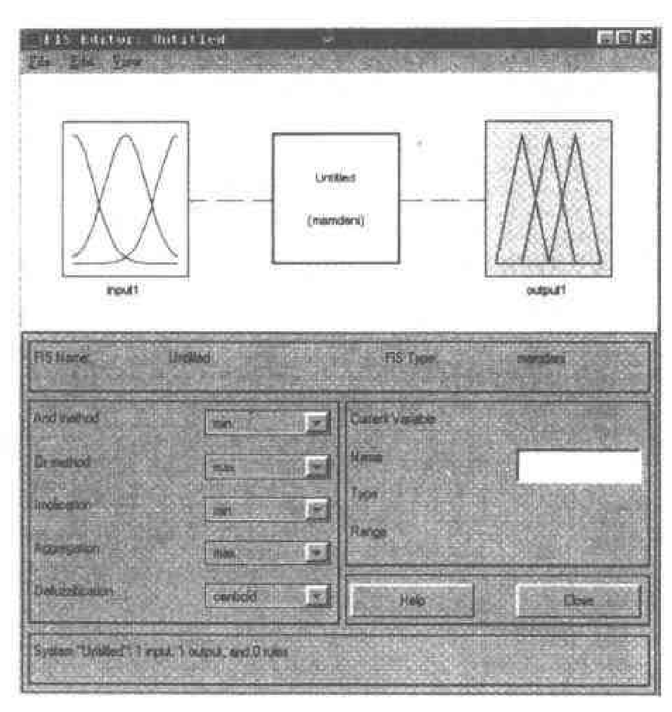


图 3-11　FIS 编辑器的图形界面

下面讲述图形界面的菜单和选项。

File 菜单

- New Mamdani FIS　　新建 Mamdani 型模糊系统。
- New Sugeno FIS　　新建 Sugero 型模糊系统。
- Open from disk ...　　打开一个保存过的模糊系统。
- Save to disk　　保存当前系统到磁盘。
- Save to disk as ...　　保存系统备份到磁盘。
- Open from workspace　　从工作空间打开保存过的模糊矩阵。
- Save to workspace　　保存模糊系统到工作空间。
- Save to workspace as　　保存系统的模糊矩阵的备份。

Edit 菜单

- Add input　　添加一个系统输入变量。
- Add outout　　添加一个系统输出变量。
- Remove variable　　移去选定的输入或输出变量。
- Undo　　撤消。

View 菜单

- Edit FIS properties　　返回定义系统的 FIS 编辑器窗口。
- Edit membership functions　　编辑选定变量的隶属函数。
- Edit rules　　打开模糊规则编辑器。
- View rules　　打开规则视图窗口，在其中可以用鼠标拖拽或直接输入来调整输入向量。这样可以观察到输出结果的表面图形和它去除离散化后的值。
- View surface　　打开表面（surface）视图窗口，可以看到随输入变量

变化的输出变量的表面图形。

下面设计一个调节一阶过程的模糊调节器来研究规则视图、表面视图和规则编辑器窗口。

考虑传递函数

$$H(z)=\frac{0.8z^{-1}}{1-0.5z^{-1}}$$

用模糊调节器来调节它，记 $y(t)$为离散输出，$r(t)$为给定信号。调节器有 2 个输入：误差 $e(t)$ 和它的变化 $\Delta e(t)$：

$$e(t)=y(t)-r(t)$$

$$\Delta e(t)=e(t)-e(t-1)$$

每个输入量，分别称为 error 和 d_error，都定义在三个模糊集上。模糊集的高斯隶属函数定义在[-10 10]区间。在由 fuzzy 命令打开的 Untitled FIS Editor 文件中，在 Edit 菜单中选择 Add Input 项添加一个输入量。用鼠标选中输入或输出变量，指定其名称或选择 max-min、sum-prod 等方法。

双击一个变量，可以选择隶属函数的个数和类型以及变量的变化范围。用鼠标选定一个隶属函数可为其指定一个名称，以便在模糊规则中应用。

图 3-12 是一个定义在区间[-10 10]上的误差变量，定义了三个高斯隶属函数，分别命名为 Negative、Nil 和 Positive。

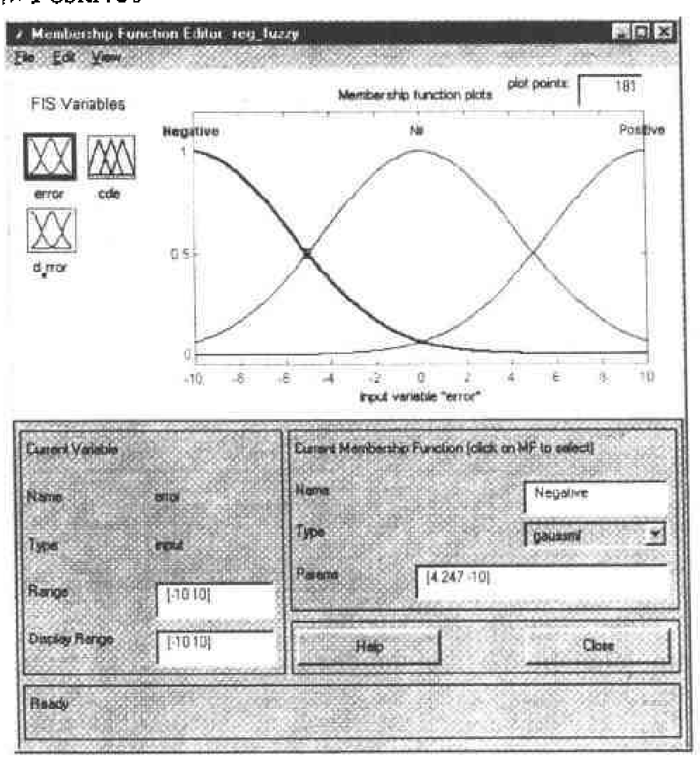


图 3-12　隶属函数编辑器

定义了全部输入和输出变量后，选择菜单 view Edit Rules 打开规则编辑窗口，如图 3-13 所示。

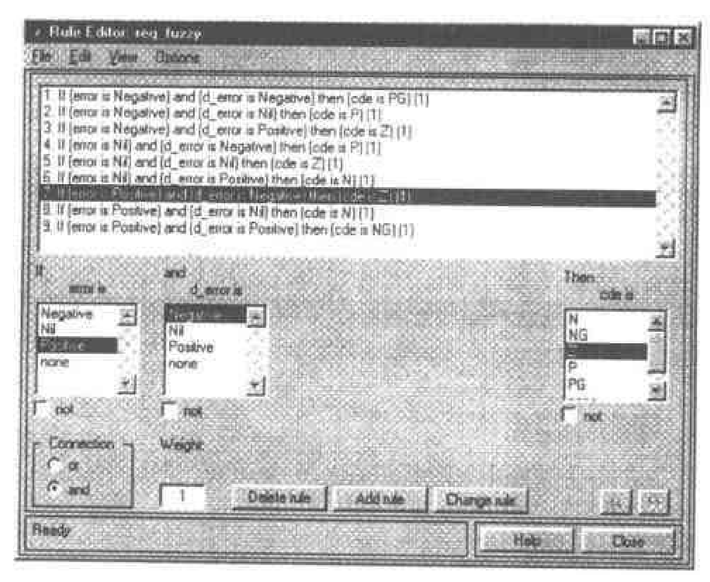


图 3-13　规则编辑窗口

在这个窗口中有一个新的菜单项 Options，可用来选择规则编写语言。每一条规则末尾有一个用括号括起来的系数。在 View 菜单中选择 Edit FIS properties 选项能够返回 FIS 编辑器窗口（见图 3-14）。

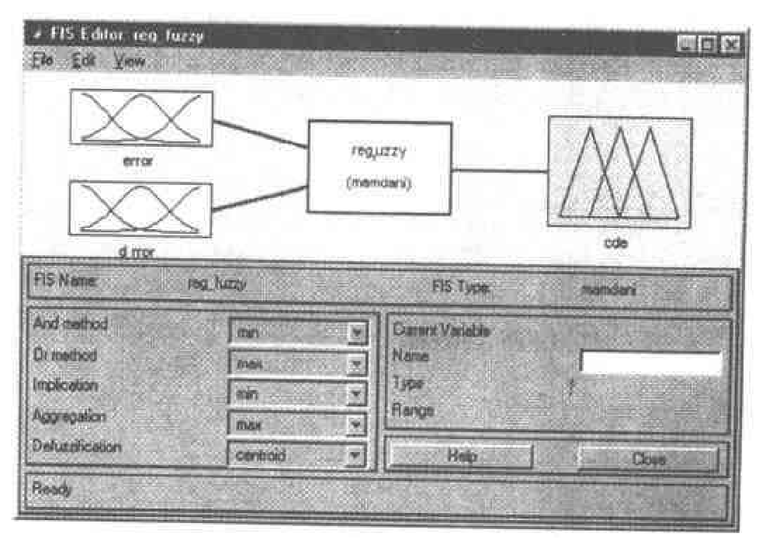


图 3-14　FIS 编辑器窗口

在本例中，系统保存在一个 FIS 模糊调节器扩展文件中。这个文件能够在集成的调试编辑器中得到调节器的文本描述。

```
》edit reg_fuzzy.fis
```

```
[System]
Name='reg_fuzzy'
Type='mamdani'
Version=2.0
NumInputs=2
NumOutputs=1
NumRules=9
AndMethod='min'
```

```
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='error'
Range=[-10 10]
NumMFs=3
MF1='Negative':'gaussmf',[4.247 -10]
MF2='Nil':'gaussmf',[4.247 0]
MF3='Positive':'gaussmf',[4.247 10]

[Input2]
Name='d_error'
Range=[-10 10]
NumMFs=3
MF1='Negative':'gaussmf',[4.247 -10]
MF2='Nil':'gaussmf',[4.247 0]
MF3='Positive':'gaussmf',[4.247 10]

[Output1]
Name='cde'
Range=[-10 10]
NumMFs=5
MF1='N':'trimf',[-10 -5 0]
MF2='NG':'trimf',[-15 -10 -5]
MF3='Z':'trimf',[-5 0 5]
MF4='P':'trimf',[0 5 10]
MF5='PG':'trimf',[5 10 15]

[Rules]
1 1, 5 (1) : 1
1 2, 4 (1) : 1
1 3, 3 (1) : 1
2 1, 4 (1) : 1
2 2, 3 (1) : 1
2 3, 1 (1) : 1
3 1, 3 (1) : 1
3 2, 1 (1) : 1
3 3, 2 (1) : 1
```

为了在 SIMULINK 模型或 M 文件中应用这个调节器，必须以矩阵的形式把它保存到工作空间。选择“Save to workspace”项保存，如果系统文件已经命名，选择“Save to workspace as” 改变模糊矩阵的名称。用户可以自己定义隶属函数、蕴含方法、合成、去除模糊化以及自己的模糊算子（AND，OR 等）。在一定条件下，用 min 替代 AND，max 替代 OR，需要较长的运算时间，因为每次只考虑一个值。只有一个隶属函数参与去除模糊化。一般的，对它们进行处理求和后用于控制过程。

3.4 用模糊工具箱命令创建模糊系统

仍以 2 个输入量（如误差和误差变化）的模糊调节器为例，每个变量都由高斯型模糊集定义。

指令 Newfis 可生成一个模糊系统，最多可有 7 个参数，它的不同语法形式有：

- syst_name　　模糊系统名称
- sys_fuzzy　　模糊系统矩阵
- syst_name　　模糊系统名称
- type　　Mamdani 或 Sugeno 型

具有 7 个参数的一般语法形式如下：

- syst_name　　模糊系统名称
- type　　Mamdani 或 Sugeno 型
- ET_method　　用于 AND 的方法
- OU_method　　用于 OR 算子的方法
- imp_method　　蕴含方法（min 或 prod）
- agg_method　　规则合成方法（max，probabilistic OR,sum）
- defuzz_method　去除模糊化方法。

下面将研究如何建立与图形界面中同样的调节器。

3.4.1 输入输出变量的模糊化

文件 prog_flou.m 中用 MATLAB 命令创建了一个同图形界面中一样的 Mamdani 型模糊调节器。用 Addvar 指令定义输入输出变量，然后由 Addmf 指令具体说明隶属函数。

```
sys_fuz = addvar (sys_fuzzy, 'type', 'name', 'interval')
```

- Sys_fuzzy　　模糊系统名称
- Type　　输入变量 input 或输出变量 output
- Nom　　模糊规则所指的变量名称
- Iuterval　　变量的取值区间

```
sys_fuz = addmf (sys_fuz, 'type', 'num', 'name', 'interva',params)
```

- sys_fuz　　模糊系统名称
- type　　输入变量 input 或输出变量 output
- nom　　模糊规则所指的变量名称
- num　　变量个数(n° 1 是第一个创建的)
- interv　　变量的取值区间
- params　　隶属函数参数（高斯函数的均值和方差等）

prog_fuzzy.m file

```
% fuzzy system created in a M-file

% creation of a new fuzzy system
```

```
sys_fuzzy = newfis('regul_fuzzy');

% input variables (the error and its variation) definition
interv_err = [-10 10];
interv_derr = [-10 10];
sys_fuzzy = addvar(sys_fuzzy,'input','err',interv_err);

% fuzzy sets of input variables
% gaussian functions with standard deviation 5, mean 0
sys_fuzzy = addmf(sys_fuzzy,'input',1,'Negative','gaussmf',...
            [5 min(interv_err)]);
sys_fuzzy = addmf(sys_fuzzy,'input',1,'Nil','gaussmf',...
            [5 mean( interv_err)]);
sys_fuzzy = addmf(sys_fuzzy,'input',1,'Positive','gaussmf',...
            [5 max(interv_err)]);
% adding the 2nd input variable: derivative of the error
sys_fuzzy = addvar(sys_fuzzy,'input','d_err',interv_derr);
% gaussian functions with standard deviation 5, mean 0
sys_fuzzy = addmf(sys_fuzzy,'input',2,'Negative','gaussmf',...
            [5 min(interv_derr)]);
sys_fuzzy = addmf(sys_fuzzy,'input',2,'Nil','gaussmf',...
            [5 mean(interv_derr)]);
sys_fuzzy = addmf(sys_fuzzy,'input',2,'Positive','gaussmf',...
            [5 max(interv_derr)]);
```

这样对模糊调节器的两个输出进行了离散化。

用指令 plotmf 绘制 sys_fuzzy 系统第一个输入量的隶属函数曲线（见图 3-15）。

```
>> plotmf ( sys_fuzzy, 'input' , 1)
>> title (' the error fuzzy sets')
>> grid
>>ylabel ('membership degrees')
```

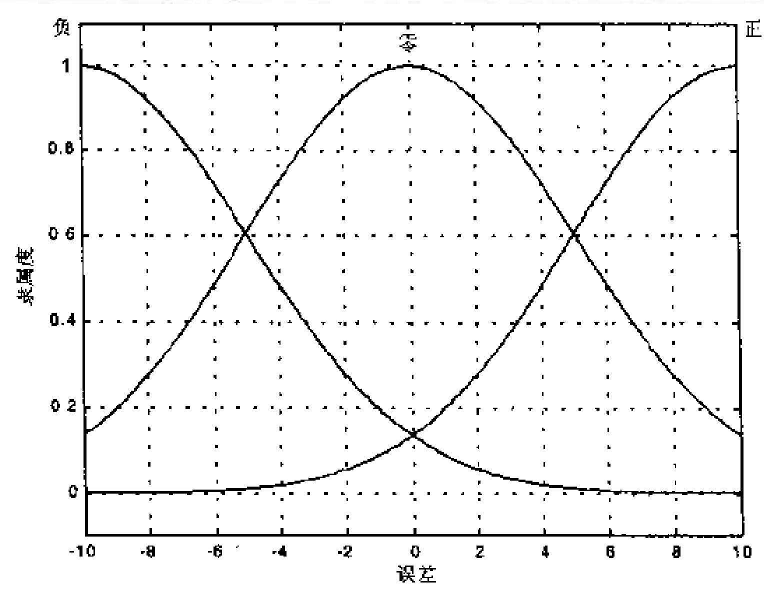


图 3-15　误差模糊集

following of prog_fuzzy.m

```
% definition of the regulator output variable
interv_cde = [-10 10];
sys_fuzzy = addvar(sys_fuzzy,'output','cde',interv_cde);

% definition of the 5 triangular membership functions
basis1 = [-15 -10 -5]; % 1st triangular membership function basis
sys_fuzzy = addmf(sys_fuzzy,'output',1,'NG','trimf',basis1);
basis = basis1+5;
sys_fuzzy = addmf(sys_fuzzy,'output',1,'N','trimf',basis);
basis = basis+5;
sys_fuzzy = addmf(sys_fuzzy,'output',1,'Z','trimf',basis);
basis = basis+5;
sys_fuzzy = addmf(sys_fuzzy,'output',1,'P','trimf',basis);
basis = basis+5;
sys_fuzzy = addmf(sys_fuzzy,'output',1,'PG','trimf',basis);
```

下面画出输出变量的隶属函数（见图 3-16）。

```
>> plotmf ( sys_fuzzy, 'output' , 1)
>> title (' Output variable  fuzzy sets')
>> grid
>>ylabel ('membership degrees')
```

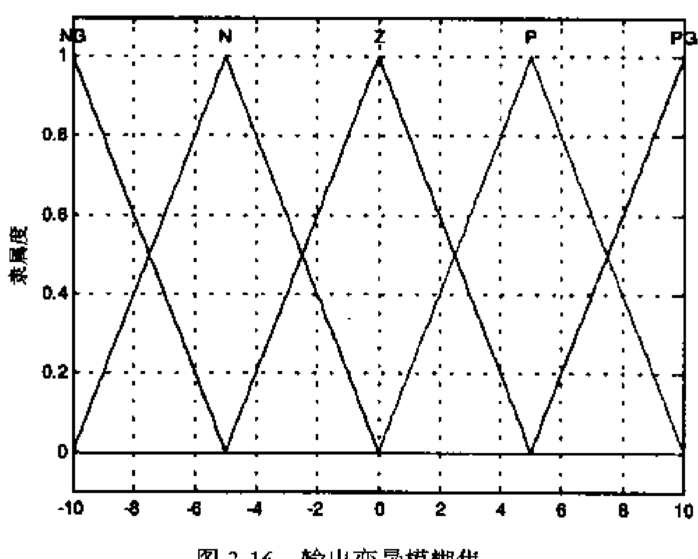


图 3-16　输出变量模糊集

我们可以移去一个输入或输出变量，或者它的一个隶属函数。

下面的指令分别为模糊系统第一个输入变量和第一个输入变量的第二个隶属函数：

$$sys_fuzzy = rm\mathrm{var}(sys_fuzzy,'input',1)$$

$$sys_fuzzy = rmmf(sys_fuzzy,'input','mf',2)$$

3.4.2　模糊规则编辑

在一个有 m 个输入量，n 个输出量的模糊系统中，所有的模糊规则可由 1 个规则矩阵定义，这个输入矩阵的行数为每个输入量的模糊集数，列数为$(m+n+2)$。第 1 个模糊规则构成矩阵的第 1 行。如果误差为负（在第 1 个模糊集 N 中），且如果它的微分也为负（在

第 1 个模糊集 N 中)，则输出是正的极大值（在第 5 个模糊集 PG 中)。这条规则的权重系数是 1。

最后一个数字是连接两个前提的算子的符号表示(1 对应 AND，2 对应 OR)（见图 3-17)。

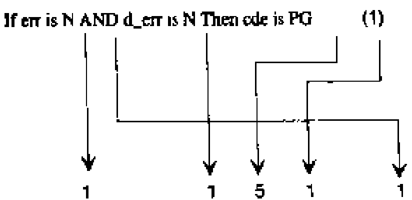


图 3-17 模糊规则的编辑示例

following of file prog_fuzzy.m

```
% matrix fuzzy rules edition
rules = [1 1 5 1 1
         1 2 4 1 1
         1 3 3 1 1
         2 1 4 1 1
         2 2 3 1 1
         2 3 2 1 1
         3 1 3 1 1
         3 2 2 1 1
         3 3 1 1 1];
sys_fuzzy = addrule(sys_fuzzy,rules);
```

模糊规则的编辑可以直接以字符串的语言形式表示。指令 parsrule 在本例中所用的语言是英语，这个指令可识别的关键字有：IF、THEN、OR、AND。只需以字符链的形式输入不同的规则，注意加空格使规则具有相同长度。

```
rules= ['if err is Negative and d_err is Negative then cde is PG ';
        'if err is Negative and d_err is Nil then cde is P ';
        'if err is Negative and d_err is Nil then cde is Z ';
        'if err is Nil and d_err is Negative then cde is P ';
        'if err is Nil and d_err is Nil then cde is Z  ';
        'if err is Nil and d_err is Positive then cde is N ';
        'if err is Positive and d_err is Negative then cde is Z ';
        'if err is Positive and d_err is Nil then cde is N  ';
        'if err is Positive and d_err is Positive then cde is NG '];
```

```
>>sys_fuzzy = parsrule (sys_fuzzy, rules, 'verbose')
```

指令 showrule 以规范的形式显示规则。

```
>>showrule(sys_fuzzy)
ans=
1If (err is Negative) and (d_err is Negative) then (cde is PG) (1)
2.If (err is Negative )and (d_err is Nil )then (cde is P) (1)
3.If (err is Negative) and (d_err is Nil )then (cde is Z) (1)
4.If (err is Nil )and (d_err is Negative )then (cde is P ) (1)
5.If (err is Nil )and (d_err is Nil) then (cde is Z) (1)
6.If (err is Nil )and (d_err is Positive) then (cde is N ) (1)
7.If (err is Positive) and (d_err is Negative )then (cde is Z ) (1)
8.If (err is Positive )and (d_err is Nil) then (cde is N) (1)
9.If (err is Positive )and (d_err is Positive) then (cde is NG) (1)
```

编写规则另外的方法还有以下三种形式：verbose、symbolic 或 indexed。可以用 ruleedit 在图形界面中打开。用指令 writefis 仍把系统保存为先前定义过的名称。

following of file prog_fuzzy.m

```
% saving on disk under the name 'regul_fuzzy.fis'
writefis(sys_fuzzy,'regul_fuzzy');
% intermediate variable suppression
clear interv_cde basis interv_derr basis1 interv_err
```

现在系统就全部定义完了。

指令 getfis 有几种语法形式，可以获得系统的信息，如输入量个数、模糊集个数等。

fuzzy set name defined by sys_fuzzy matrix

```
>> getfis (sys_fuzzy ,'name')
ans =
reg_fuzzy
```

◆ 第一输入特征

```
>> getfis (sys_fuzzy ,'input',1)
Name =   err
NumMFs = 3
MFLabels =
       Negative
       Nil
       Positive
Range =     [-10 10]
```

◆ 第二输入特征

```
>> getfis (sys_flou ,'input',2)
Name =   d_err
NumMFs =  3
MFLabels =
      Negative
      Nil
      Positive
Range =     [-10 10]
```

◆ 输出特征

```
>> getfis (sys_flou ,'output',1)
Name =   cde
NumMFs =  5
MFLabels =
       NG
       N
       Z
       P
      PG
Range =     [-10 10]
```

◆ 模糊系统特征

```
>>getfis(sys_fuzzy)
Name    =  regul_fuzzy
Type    =  mamdani
NumInputs = 2
InLabels  =
```

```
        err
        d_err
Outputs =1
OutLabels = cde
NumRules = 9
AndMethod = min
OrMethod = max
ImpMethod = min
AggMethod = max
DefuzzMethod = centroid
ans = regul_fuzzy
```

◆ 第一输入变量之第一模糊集特征

```
>>getfis(sys_fuzzy,'input',1,'mf',1)
Name = Negative
Type = gausmf
Params = [5  -10]
```

上面的指令得出了第一个输入量的第一个隶属函数"Negative"，它是标准偏差和均值分别为5和10的高斯型。

◆ 模糊系统特征

指令 showfis 完全定义了模糊系统的每个输入输出，并返回所有的参数（名称、取值范围、个数和隶属函数类型等），而且它还表明所用的方法和显示规则列表。

```
>>showfis(sys_fuzzy)
1. Name                  regul_fuzzy
2. Type                  mamdani
3. Inputs/Outputs        [2 1]
4. NumInputMFs           [3 3]
5. NumOutputMFs          5
6. NumRules              9
7. AndMethod             min
8. OrMethod              max
9. ImpMethod             min
10. AggMethod            max
11. DefuzzMethod             centroid
12. InLabels                 err
13.                      d_err
14. OutLabels                cde
15. InRange                  [-10  10]
16.                      [-10  10]
17. OutRange                 [-10  10]
18. InMFLabels               Negative
19.                      Nil
20.                      Positive
21.                      Negative
22.                          Nil
23.                          Positive
24. OutMFLabels              NG
                             N
                             Z
                             P
```

```
                               PG
29. InMFTypes                  gaussmf
30.                            gaussmf
31.                            gaussmf
32.                            gaussmf
33.                            gaussmf
34.                            gaussmf
35. OutMFTypes                 trimf
36.                            trimf
37.                            trimf
38.                            trimf
39.                            trimf
40. InMFParam                  [5  -10  0  0]
41.                            [5   0   0  0]
42.                            [5   10  0  0]
43.                            [5  -10  0  0]
44.                            [5   0  0  0]
45.                            [5   10  0  0]
46. OutMFParams                [-15  -10  -5  0]
47.                            [-10   -5   0  0]
48.                            [-5   0  5  0]
49.                            [0   5  10  0]
50.                            [5   10  15  0]
51.RulesList                   [1  1  5  1  1]
52.                            [1  2  4  1  1]
53.                            [1  3  3  1  1]
54.                            [2  1  4  1  1]
55.                            [2  2  3  1  1]
56.                            [ 2  3  2  1  1]
57.                            [ 3 1  3  1  1]
58.                            [3  2  2  1  1]
59.                            [3  3  1  1  1]
```

◆ 规则显示

指令 showrule 没有具体的语言限制，可以以任意一种语言形式显示规则列表。

```
>>showrule(sys_fuzzy)
1.If (err is Negative) and (d_err is Negative) then (cde is PG) (1)
2.If (err is Negative )and (d_err is Nil )then (cde is P) (1)
3.If (err is Negative) and (d_err is Nil )then (cde is Z) (1)
4.If (err is Nil )and (d_err is Negative )then (cde is P ) (1)
5.If (err is Nil )and (d_err is Nil) then (cde is Z) (1)
6.If (err is Nil )and (d_err is Positive) then (cde is N ) (1)
7.If (err is Positive) and (d_err is Negative )then (cde is Z ) (1)
8.If (err is Positive )and (d_err is Nil) then (cde is N) (1)
9.If (err is Positive )and (d_err is Positive) then (cde is NG) (1)
```

用另外的形式，如 symbolic，也许只能显示一些规则。下面的指令以 symbolic 形式，按顺序显示规则 1、4、3、9、7 和 2。

```
>>disp_rules = showrule (sys_fuzzy, [1 4 3 9 7 2])
disp_rules =
```

◆ 由输入变量产生输出变量的表面

指令 sufview(sys_fuzzy) 打开表面（surface）视图窗口，用鼠标在其中拖动任意点可以改变视角。为了获得更好的图形精度，也可以通过输入更大的 Xgrids 和 Ygrids 值来增加每个输入量的点数。指令 gensurf(sys_fuzzy)在一个图形窗口画出这一表面（见图 3-18）。

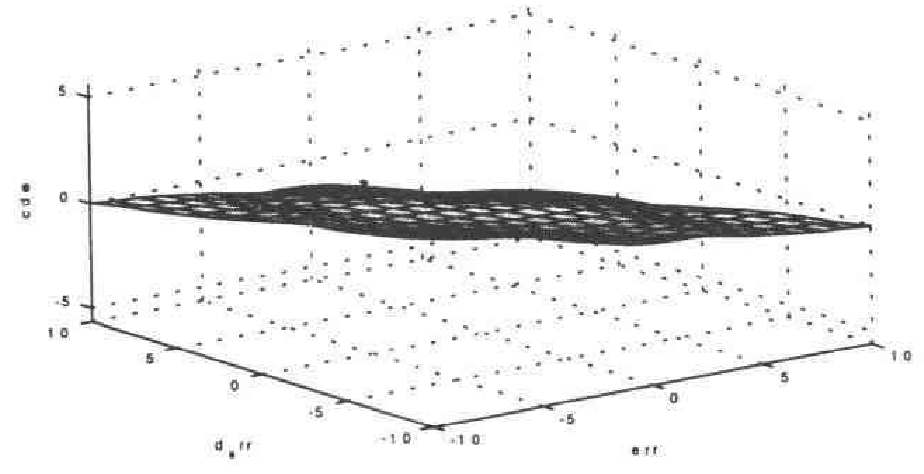


图 3-18 模糊系统的三维图形表面

由 gensurf 得到的表面视角可以通过在指令 view 中指定适当的方位角和高度来改变。

```
>> AZ = 45 ; EL = 30; gensurf(sys_fuzzy), view(AZ,EL)
>>tiltle('control signal according to the error and its variation')
```

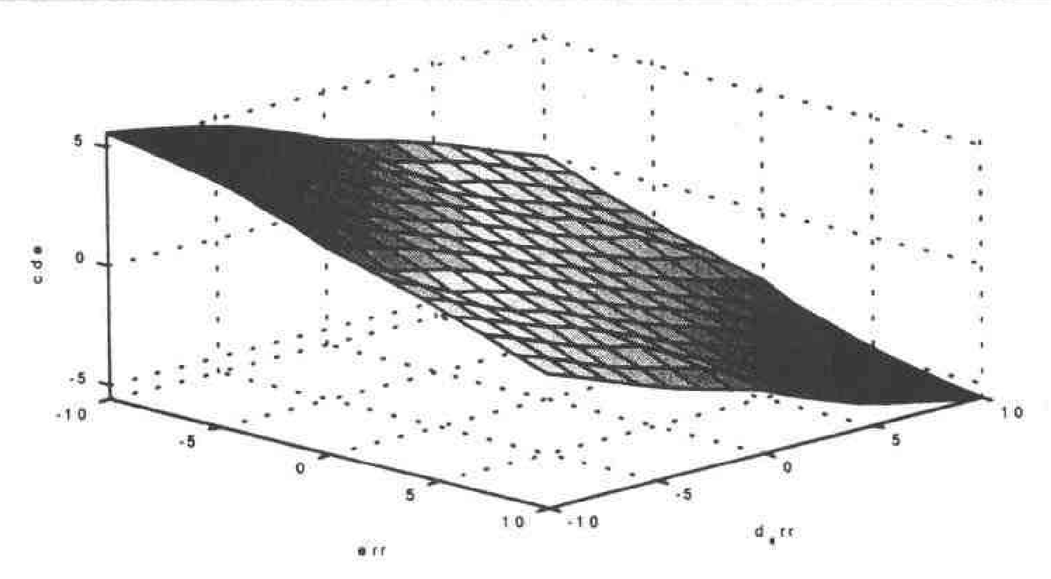


图 3-19 根据误差和它的差分的控制信号

图 3-19 在指令 view 中指定适当的方位角和高度得到的模糊系统 3 维图形表面系统的图形表示由 plotfis 获得，它可以简单明了地表示模糊系统的主要参数（输入、输出变量的个数，隶属函数的个数和类型等）。

```
>>plotfis(sys_fuzzy)
>>gtext ('fuzzy system's graphic representation)
```

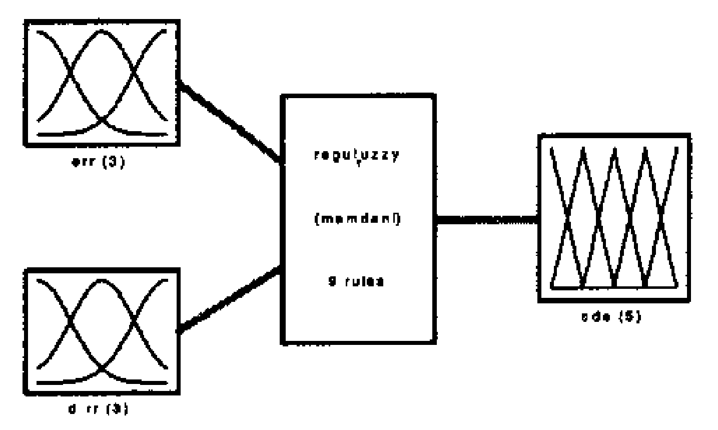


图 3-20　模糊系统的图形化表示

3.4.3　去除模糊化

指令 ruleview 打开规则视图窗口，在其中可以观察到运用 max-min 方法的去除离散化。用鼠标选中输入量的任意值，并查看用 max-min 法得到的输出变量的隶属函数。

对于误差为 0.4545、其导数为 7.727 的情况，调节器产生的控制信号为−2.6528。这个结果可由指令 evalfis 得到。如果模糊调节器的矩阵不在工作空间，可用指令 readfis 打开。

```
>>sys_fuzzy = readfis ('regul_fuzzy');
>> x =[0.4545 7.7271];
>> y = evalfis(x,sys_fuzzy)
y =
   -2.6528
```

```
>>ruleview(sys_fuzzy)
```

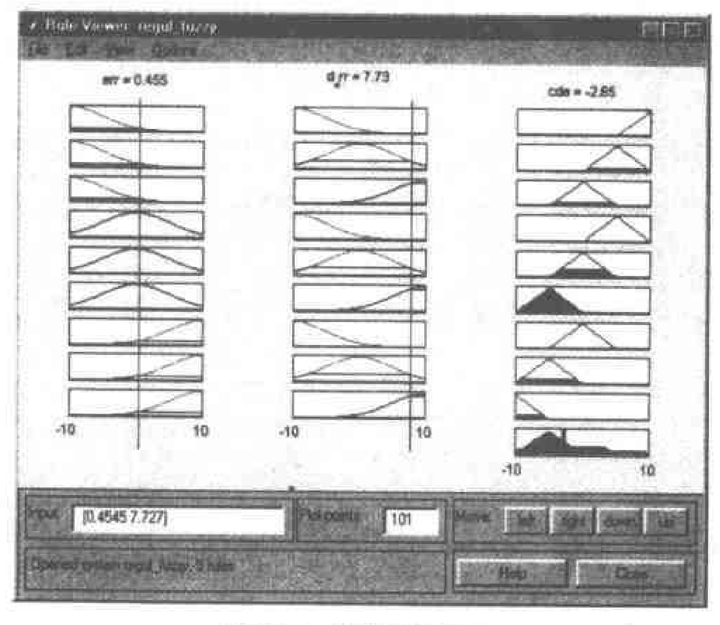


图 3-21　规则视图窗口

3.4.4 在控制律中应用调节器

为应用这个调节器，必须运行 prog_fuzzy.m 文件，它定义了模糊系统。

ctr_fuzzy.m file

```
% Using the regulator in a control law
close all

% definition of the fuzzy regulator
prog_fuzzy2

% target signal
r = (0:60)/12; triangle = [r fliplr(r)];
square_signal = [zeros(1,100) 5*ones(1,100) zeros(1,100)];
triangle = triangle+sin(0.2*( 0:length(triangle)-1));
r = 2+[triangle square_signal];

% control signal initialisation
u = ones(1,2);
y = r; err = zeros(1:2);
Gain = 0.3;
% Gain = 1.8;
% Gain=0.8;
for i = 3:length(r)-1
   % process signal output
     y(i) = 0.7*y(i-1)+0.3*u(i-1)+0.05*u(i-2);

     % error and its derivative
     err(i) = y(i)-r(i+1);
     d_err(i) = err(i)-err(i-1);

     % control signal
     x = [err(i) d_err(i)];
     du(i) = evalfis(x,sys_fuzzy);
     u(i) = u(i-1)+Gain*du(i);

end
% displaying the results

% target signal
plot(r,':'), axis([0 length(r) 0 9]), hold on

% process signal output
plot(y)
grid
title1 = 'fuzzy logic control - ';
title2 = ' target and output signals';
title(strcat(title1,title2));
xlabel('discrete time')
figure(2)
% control signal to be applied to the process
stairs(u)
```

```
xlabel('discrete time')
grid
title('applied control signal')
figure(3)
% fuzzy regulator's output signal
stairs(du);
title('output signal of the regulator')
xlabel('discrete time'), grid
figure(4)
subplot(211)
% error signal
plot(err)
axis([0 length(err) -6 6])
title('error')
grid
subplot(212)
% error variation
plot(d_err), axis([0 length(d_err) -6 6])
title('error derivative'), grid
xlabel('discrete time')
```

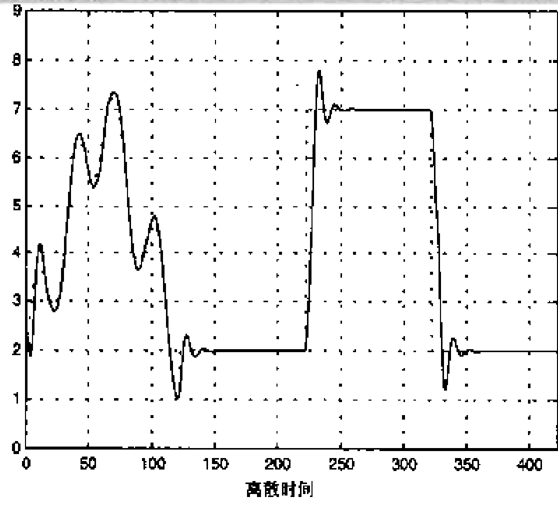


图 3-22 模糊逻辑控制系统的目标信号和输出信号

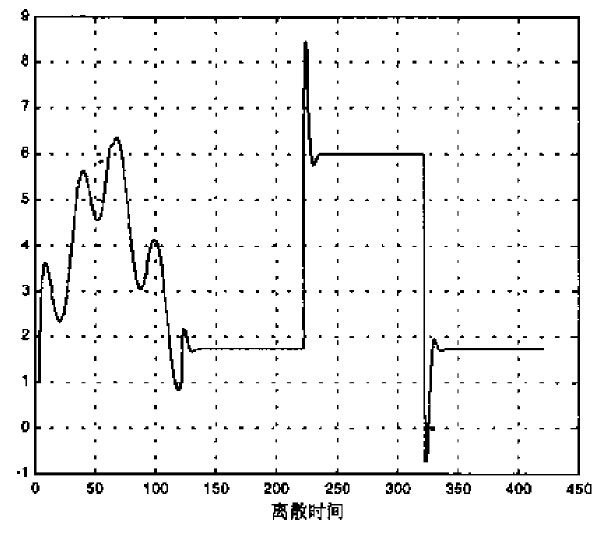


图 3-23 应用的控制信号

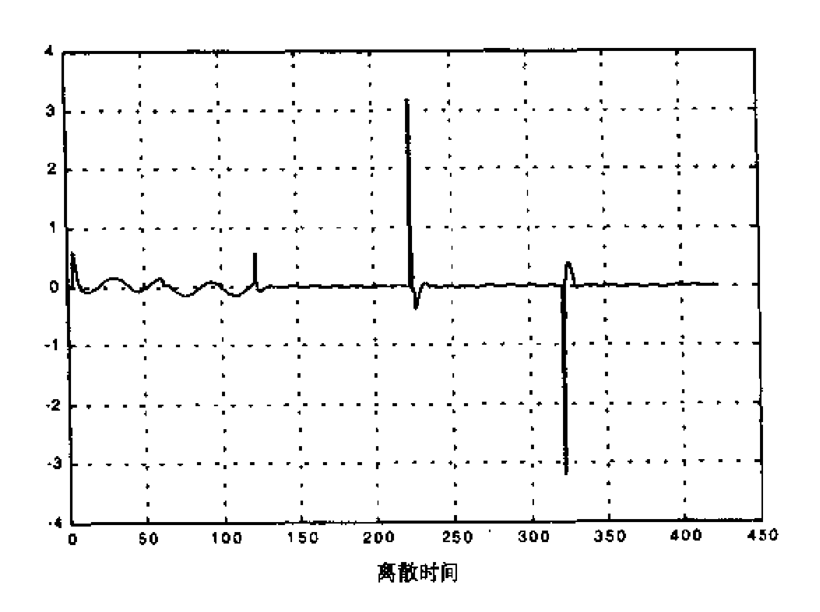


图 3-24　调节器的输出信号

可以验证，由于积分环节消除了稳态误差，稳态时模糊调节器产生的控制信号等于零。从误差和它的微分曲线很容易得出推理表（模糊规则）（见图 3-25、图 3-26）。

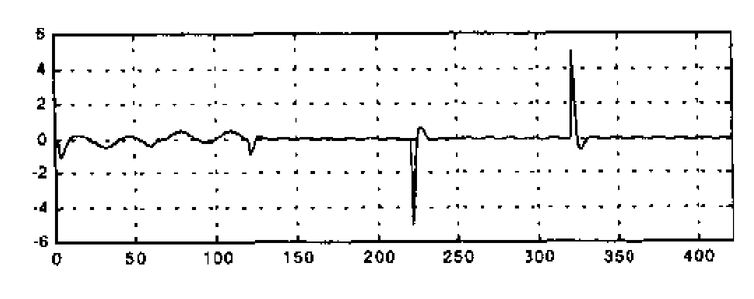


图 3-25　误差变化曲线

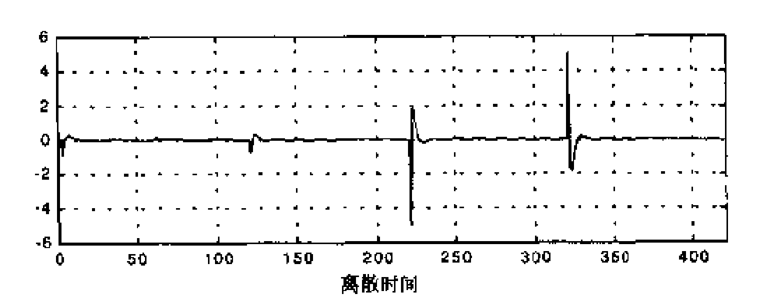


图 3-26　误差微分变化曲线

在控制律中增加积分环节可以使稳态误差为零，事实上是因为模糊调节器产生的是一个控制信号的变化量，它在与前一时刻的控制量相加后才应用到过程。增益为 0.8 时，可以观察到随目标信号变化的误差（见图 3-27）。

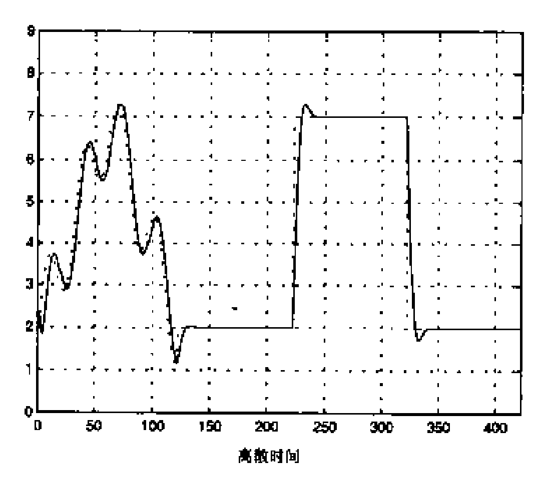


图 3-27 模糊逻辑控制系统的目标信号和输出信号

在模糊化阶段减小误差和它的导数的取值范围可以使控制系统更加动态化。

文件 prog_fuzzy2.m 的命令行改变了误差的定义区间，从而误差隶属函数的参数（均值和标准偏差）也发生了改变。

```
% Input variables (the error and its variation) definition
interv_derr = [-10 10];

% fuzzy sets of input variables
% input variable definition and its variation
interv_err = [-1 1];
sys_fuzzy = addvar(sys_fuzzy,'input','err',interv_err);

% input variables fuzzy sets definition

% gaussian membership functions with standard deviation = 1,
% and respectively means of -1, 0 et 1
sigma = 1;
sys_fuzzy = addmf(sys_fuzzy,'input',1,'Negative','gaussmf',...
            [sigma min(interv_err)]);
sys_fuzzy = addmf(sys_fuzzy,'input',1,'Nil','gaussmf',...
            [sigma mean(interv_err)]);
sys_fuzzy = addmf(sys_fuzzy,'input',1,'Positive','gaussmf',...
            [sigma max(interv_err)]);
```

在文件 ctr_fuzzy 中引用 prog_fuzzy2.m 定义的模糊系统。增益为 0.3 时有如下结果（见图 3-28）。

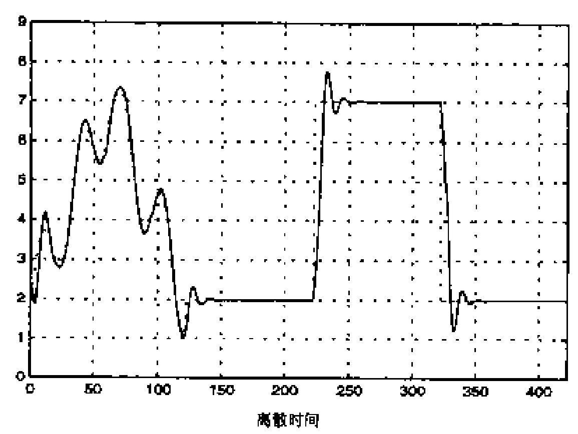


图 3-28 模糊逻辑控制系统的目标信号和输出信号

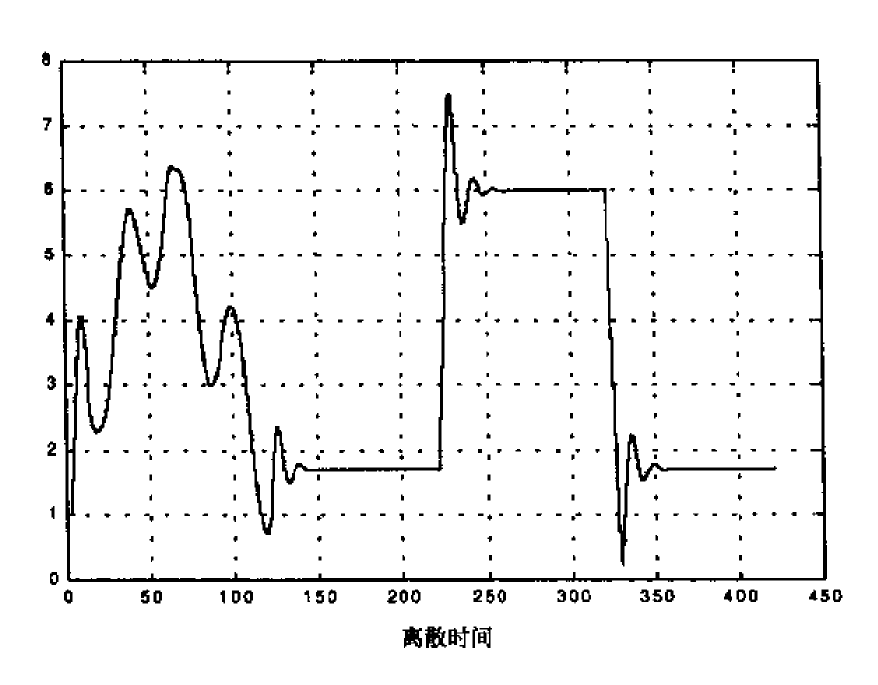


图 3-29 应用的控制信号

闭环系统的动态特性可由输入输出变量的定义决定，或者由增益系数决定。假定无先验知识时，增益系数使误差和它的导数的定义区间有一定的自由度（见图 3-30 至图 3-32）。

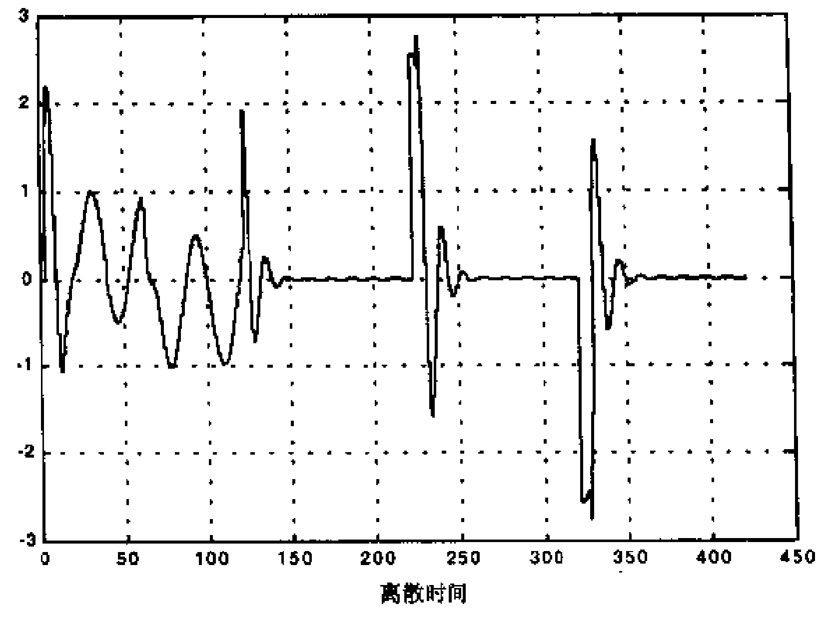


图 3-30 调节器的输出信号

图 3-31 误差变化曲线

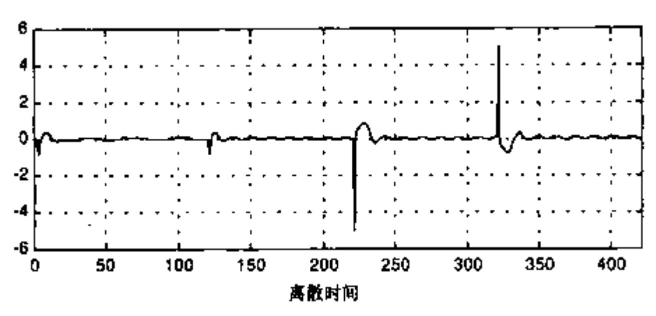


图 3-32　误差微分变化曲线

3.5　在 SIMULINK 中应用模糊调节器

下面的模块表示 Blocksets & Toolboxes/SIMULINK®Fuzzy 中的一个模糊系统（见图 3-33）。



图 3-33　SIMULINK 中的模糊系统模块

在图形界面中，已经用名称 regul_fuzzy.fis 在工作空间以模糊矩阵 regul_fuzzy 的形式保存了这个系统。为把这个单元和前面创建的模糊调节器联系起来，双击图标，在工作空间输入矩阵的模糊调节器名（见图 3-34）。

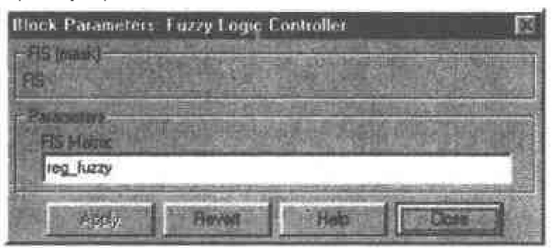


图 3-34　模糊系统模块参数输入窗口

选择 Edit 菜单的 Edit Mask 选项，打开下面的窗口，表明 MATLAB 调用 S 函数 sffis，把表示模糊系统的矩阵作为参数传给它（见图 3-35）。

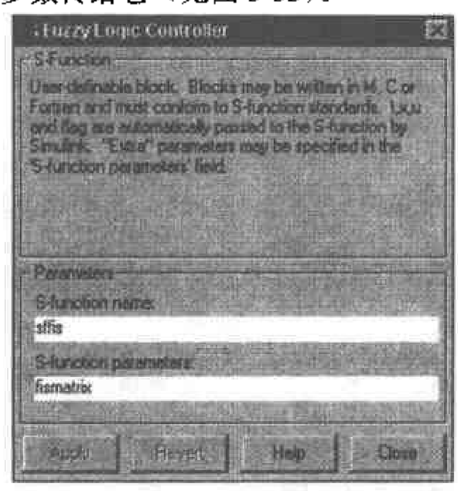


图 3-35　调用 S 函数 sffis 传递模糊系统矩阵

由于调节器有 2 个输入——误差及其导数，这两个量必须是多路复用。过程的控制量等于调节器的输出加上前一个采样时刻的控制量。给定信号和输出信号是多路复用，以便同时显示在一个范围（scope）窗口并存储在 cs.mat 文件中。由于转换单元，调节器的输出信号和过程的整个控制具有同样的特征。模糊控制的增益在 t=300 以前是 0.2，以后是 0.8。

如图 3-36 所示，双击转换单元，在其中指定输入信号端口改变的时刻（从端口 1 到端口 2）。

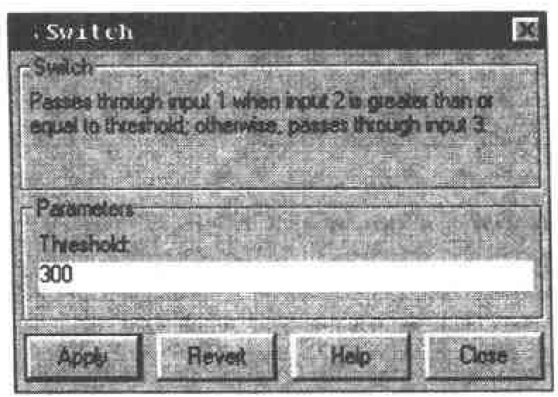


图 3-36　转换单元

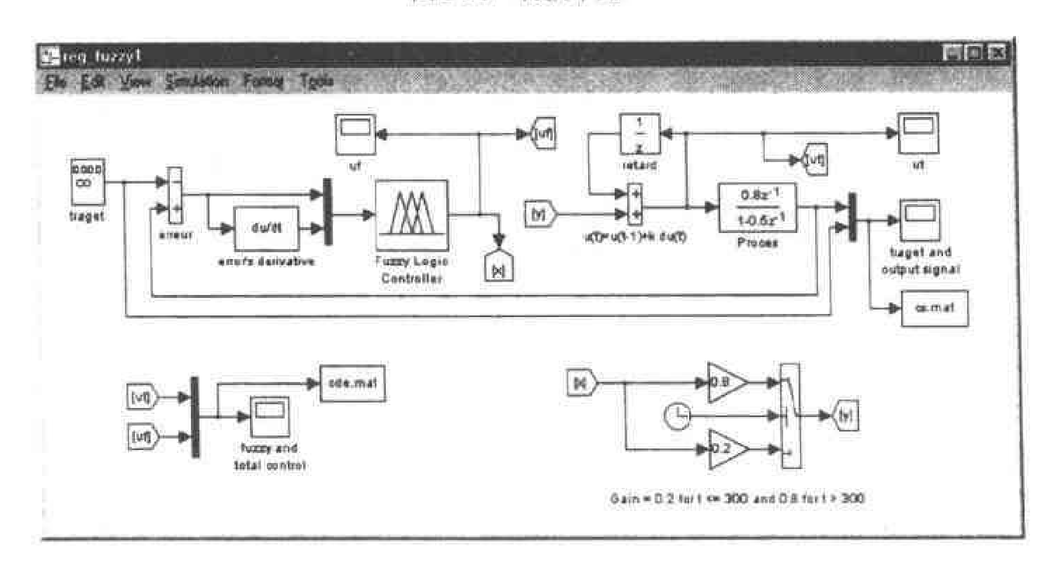


图 3-37　**SIMULINK 中的模糊逻辑系统**

read_fic.m

```
% fuzzy regulator

% reading of input/output signal and displaying the results
load cde.mat
load cs.mat
t = cs(1,:);
ut = cde(2,:);  % applied control to the process
uf = cde(3,:);  % fuzzy regulator's output
y = cs(2,:);    % process output
r = cs(3,:);    % target signal
% displaying of the target and process output signals
figure(1)
```

```
plot(t,r), hold on
h = plot(t,y);
set(h,'LineWidth',2);
plot(300*ones(1,25),-1.2:0.1:1.2,'-.')
axis([0 600 -1.5 1.5]), hold off
title('target and process output signals')
xlabel('discrete time')
text(150,-1.25,'Gain k = 0.2')
text(400,-1.25,'Gain k = 0.8')

% plotting the control signal
figure(2), stairs(t,ut), hold on

% plotting the regulator's output signal
h = stairs(t,uf);
set(h,'LineWidth',2);
plot(300*ones(1,21),-1:0.1:1,'-.'), hold off
title('fuzzy regulator''s output and the applied control')
xlabel('discrete time')
text(150,-1.25,'Gain k = 0.2')
text(400,-1.25,'Gain k = 0.8')
```

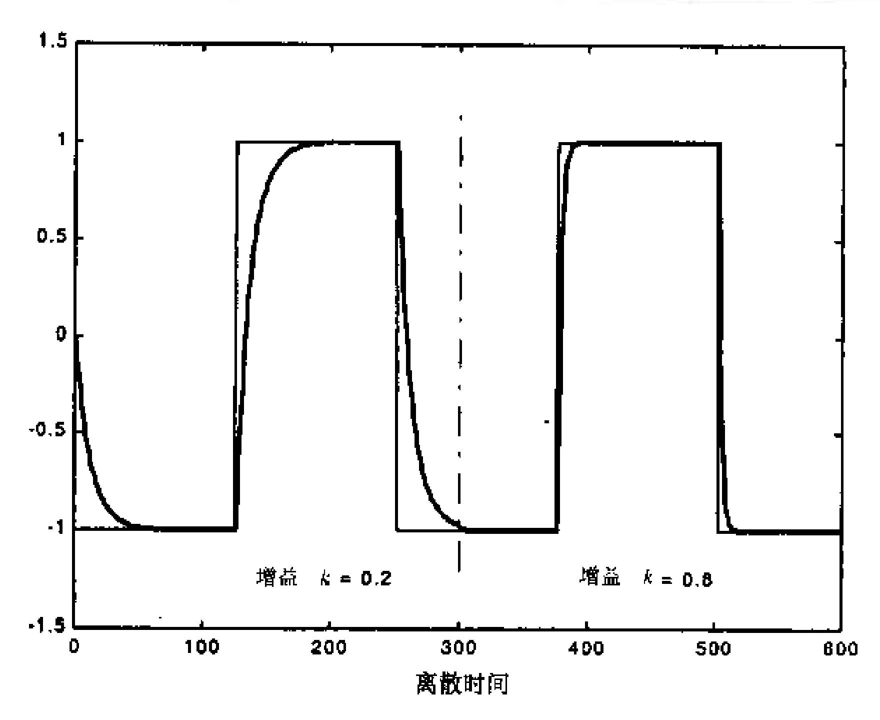


图 3-38　目标信号和过程输出信号

这样的模糊控制类似于一个 PI（比例积分器）。积分消除了稳态误差而增益影响响应时间。

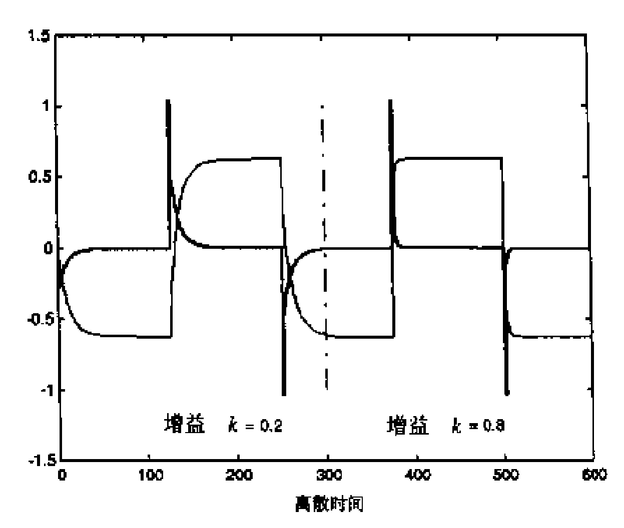


图 3-39 模糊调节器的输出和应用的控制信号

对于一个给定的正弦信号，增益 0.8 能减小跟踪误差（见图 3-40)。

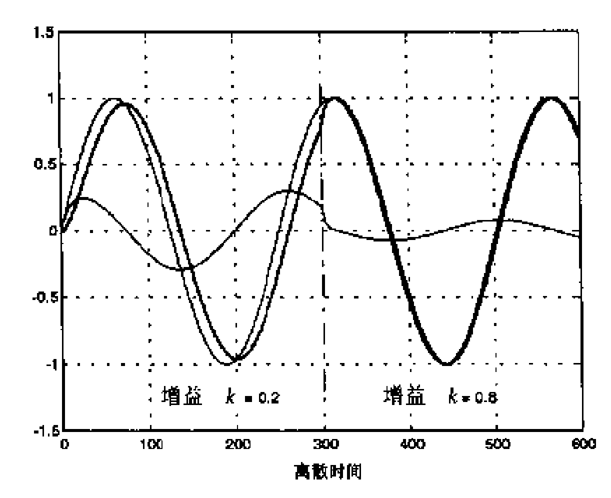


图 3-40 目标信号、过程输出和误差信号

3.6 Sugeno 方法

Mamdani 方法可追溯到 1975 年，在过程控制理论中的应用非常广泛。Sugeno 方法的不同之处在于输出变量的定义及相应的去除模糊化的方法。Sugeno 方法的输入模糊化和 Mamdani 方法没有什么不同。输出变量或者取一个不依赖输入的常数（标量），或者是输入变量的线性组合。这个标量在去除模糊化阶段由输入变量的隶属度加权。对于一个两输入量的系统，Sugeno 型的一般规则如下：

```
If e1 is A AND e2 is B then output =p*e1+q*e2+r
```

其中 A 和 B 分别代表 e1、e2 的隶属函数。系数 p、q 和 r 由用户选择，决定输入量的线性组合。在图形界面中，这些常量由 membership functions editor 中的行向量 Params 定义，选择标量相当于使 p 和 q 的各个元素为零。

3.6.1 用图形界面实现模糊调节器

从 file 菜单中选择 edit from disk 项，在图形界面中打开文件 reg_sug.fis 定义的调节器。

它的特征可由指令 Edit 以文本形式显示。

下面的命令列出了定义误差和它的导数的模糊集的隶属函数。

```
% reading the fuzzy system
fismat = readfis ('reg_sug');

% displaying the membership function of the 1st input variable

plotmf(fismat,'input',1)
title('error' 's fuzzy sets')
ylabel('Sugeno' 'regulator')
grid

% displaying the membership function of the 2nd input variable

figure(2)
plotmf(fismat, 'input',2)
title('error' 's variation fuzzy sets')
ylabel('Sugeno' 'regulator')
grid
```

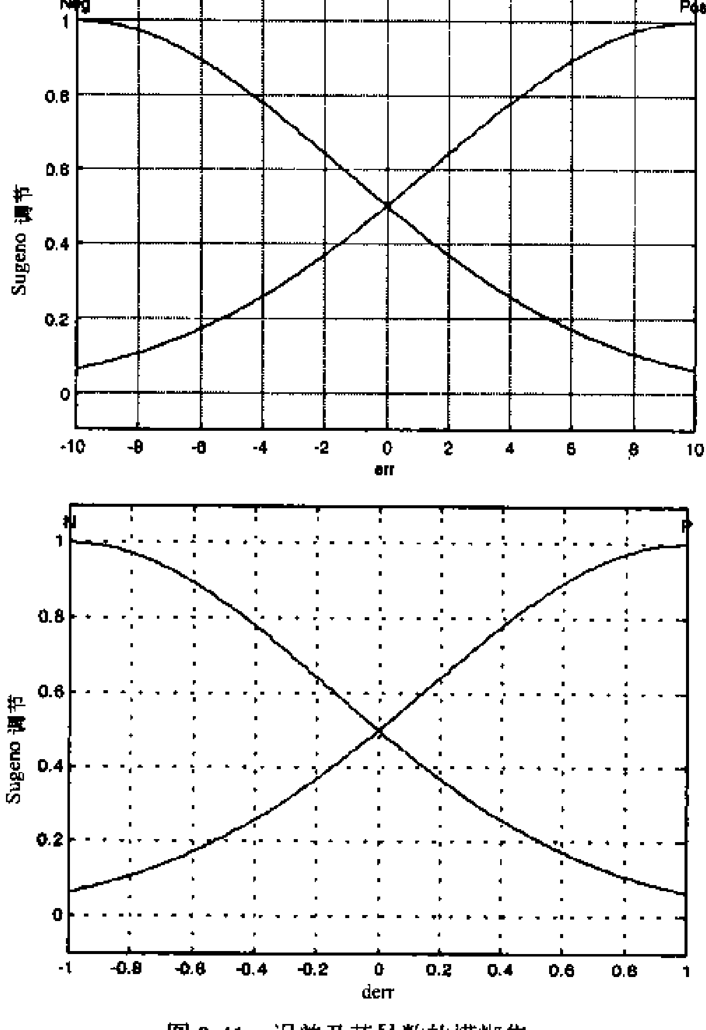


图 3-41　误差及其导数的模糊集

由于每个输入量有 2 个模糊集，所以输出变量定义在 4 个模糊集上，如表 3-2 所示。

表 3-2　推理表

$\dot{\varepsilon}$ \ ε	N	P
N	NN	NP
P	PN	PP

在每种情况下，输出都是输入的线性组合。

NN 或 PP：

```
cde = (2)*err + (1)*d_err + (0), 其中 Param = [2 1 0]
```

NP 或 PN：

```
cde = (1)*err + (1)*d_err + (0), 其中 Param = [1 1 0]
```

注意：params 向量长度等于输入个数+1。我们可以为输出量的每个模糊集选一个常量，并用最小隶属度加权（在 AND 情况下）。

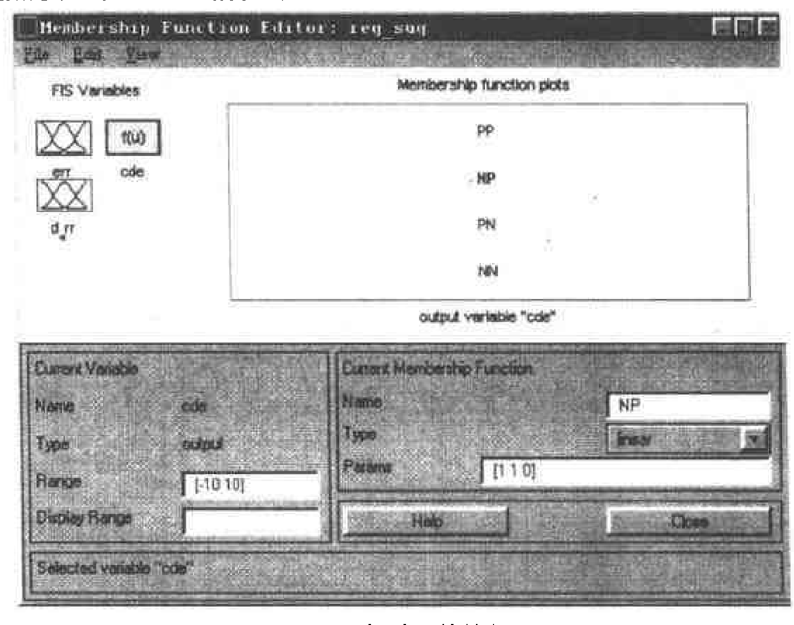


图 3-42　隶属函数编辑器

由一个输入量的模糊集隶属度可知，输出量是它们的线性组合。例如，使误差及其导数分别为 1 和−0.5。下面的命令行计算每个变量在不同模糊集上的隶属度。结果如图 3-43 所示。

```
input = [1 -0.5];
errN = evalmf (input(1), [8.5 -10], 'gaussmf')
errP = evalmf (input(1), [8.5 10], 'gaussmf')
d_errN = evalmf (input(2), [0.85 -1], 'gaussmf')
d_errP = evalmf (input(2), [0.85 1], 'gaussmf')
errN =
    0.4328
errP =
    5709
d_errN =
```

```
   8411
d_errP =
   0.2107
```

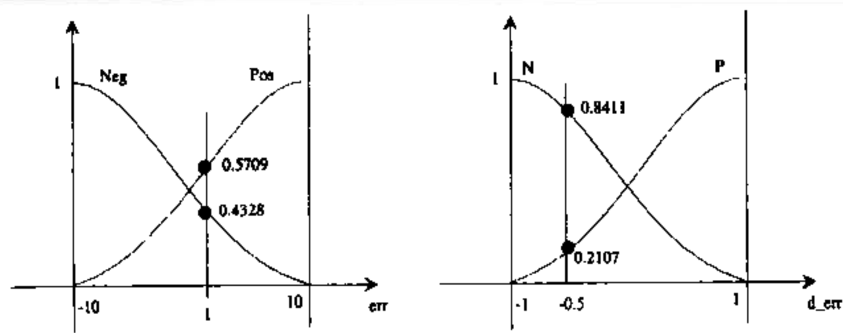


图 3-43 计算结果

模糊逻辑工具箱中有两种不同的 Sugeno 方法：wtsum 和 wtaver。输出常量取决于输入量。在由隶属度加权以前，其值如下：

NN 或 PP：

```
cde = (2)*1 + (1)*(-0.5) = 1.5
```

PN 或 NP：

```
cde = (1)*1 + (1)*(-0.5) = 0.5
```

在每条规则中包含 AND 算子时，由 min 法得出如下加权系数（见图 3-44）。

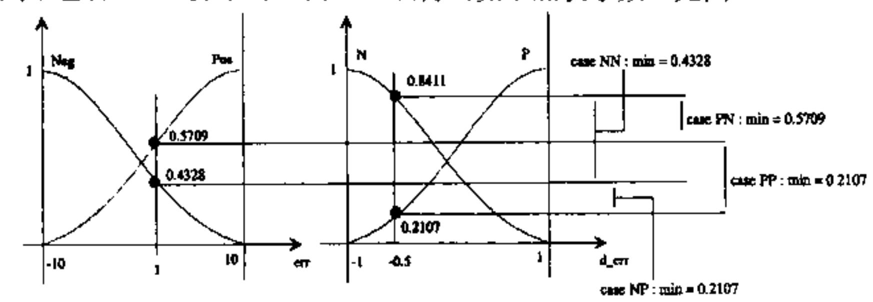


图 3-44 计算结果

每种情形下，标量的值都由隶属度的最小值加权。

```
case NN: cde =1.5*0.4328 = 0.6492
case NP: cde =0.5*0.2107 = 0.1054
case PN: cde =0.5*0.5709 = 0.2854
case PP: cde =1.5*0.2107 = 0.3160
```

用 wtsum 法时，最后的标量值等于加权后各项的和：

```
cde =0.6492+0.1054+0.2854+0.3160 = 1.3560
```

若打开规则视图窗口可得到同样的输出值（见图 3-45）。

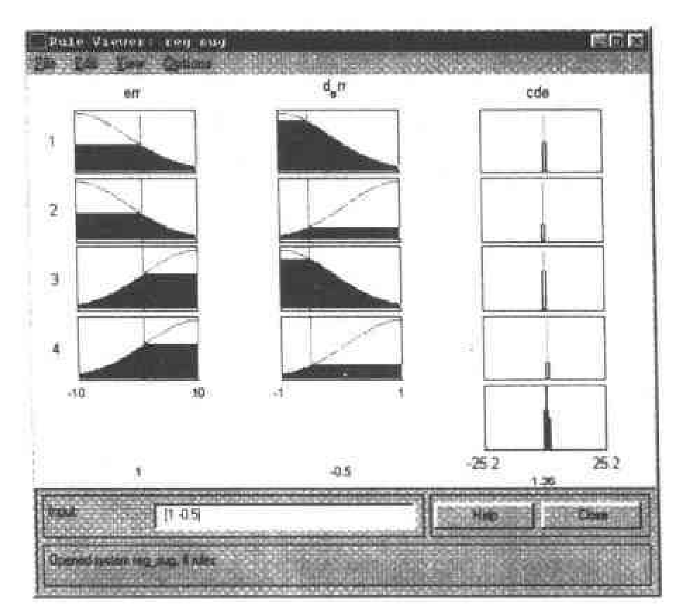


图 3-45　规则视图窗口

在 cde 列，窄线和粗线分别代表加权前后的常量。如果用 wtaver 方法，用 wtsum 的结果除以隶属度的和，结果如下：

$$cde=-\frac{1.5*0.4328+0.5*0.2107+0.5*0.5709+1.5*0.2107}{0.4328+0.2107+0.5709+0.2107}=0.9515$$

图 3-46 为对应于输入的输出变量的表面图。

```
>>fismat = readfis('ref_sug');
>>gensurf(fismat)
>>title('suface generated by the regulator output')
```

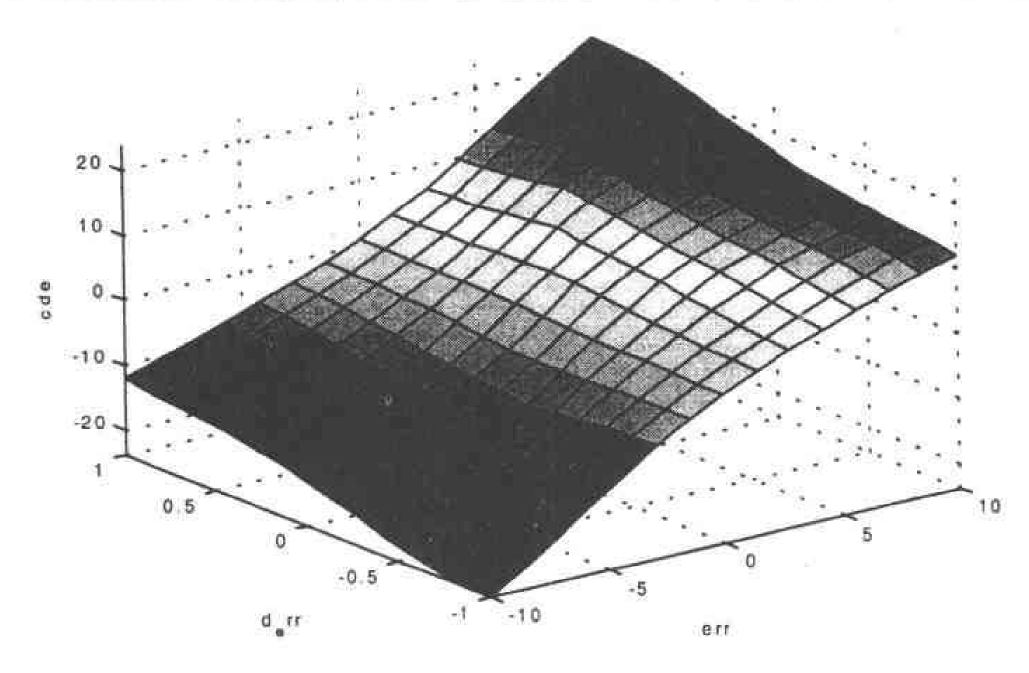


图 3-46　调节器输出量的表面

在 cde_sug.m 文件中，调节器用于和前面提到的同样的给定信号的跟踪。t=150 时，加入干扰信号 1。

cde_sug.m file

```
% Use of the Sugeno's regulator in a control law
% reading the fuzzy matrix of the fuzzy regulator
reg = readfis('reg_sug');
% target signal
r = (0:60)/12 ; triangle = [r fliplr(r)];
carre = [zeros(1,100) 5*ones(1,100) zeros(1,100)];
triangle = triangle+sin(0.2*( 0:length(triangle)-1));
r = 2+[triangle carre];
% command initialisation
u = ones(1,2); y = r; err = zeros(1:2);
for i = 3:length(r)-1
     % reading the process output
     y(i) = 0.8*y(i-1)+u(i-1)-0.5*u(i-2);

     % error and its derivative
     err(i) = r(i+1)-y(i);
     d_err(i) = err(i)-err(i-1);
     % regulator's input
     x = [err(i) d_err(i)];
     % regulator output
     du(i) = evalfis(x,reg);
     % applied control signal
     u(i) = u(i-1)+0.2*du(i);
     % disturbance at t = 150
     y(i) = y(i)+(i == 150);
end

% displaying results
plot(r,':')
axis([0 length(r) 0 9])
hold on, plot(y), grid
title1 = 'fuzzy regulation, ';
title2 = ' target and process output signals';
title(strcat(title1,title2));
xlabel('discrete time'), ylabel('Sugeno''s method')
figure(2), stairs(u)
xlabel('discrete time'), grid
title('applied control signal');
figure(3), stairs(du);
title('fuzzy regulator output')
xlabel('discrete time'), grid
figure(4), subplot(211), plot(err)
axis([0 length(err) -6 6])
title('error')
grid, subplot(212)
plot(d_err)
axis([0 length(d_err) -6 6])
title('derivative of the error'), grid
xlabel('discrete time')
```

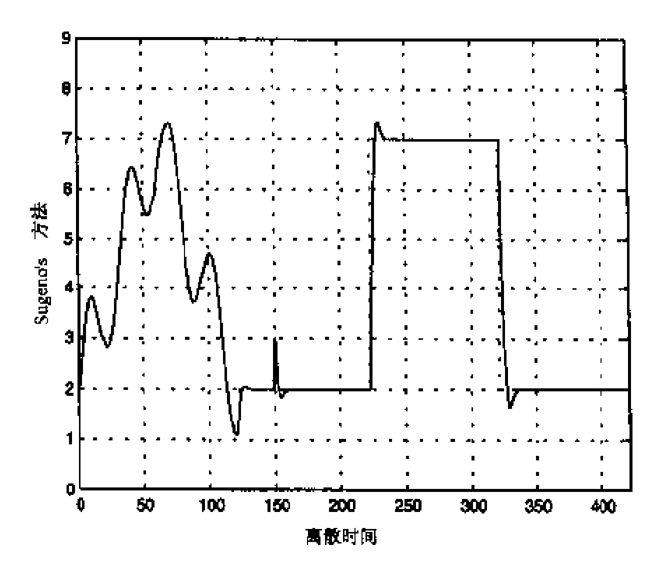


图 3-47　模糊调节器的目标信号和过程输出信号

t=150 时，干扰信号被抑制，没有产生超调。

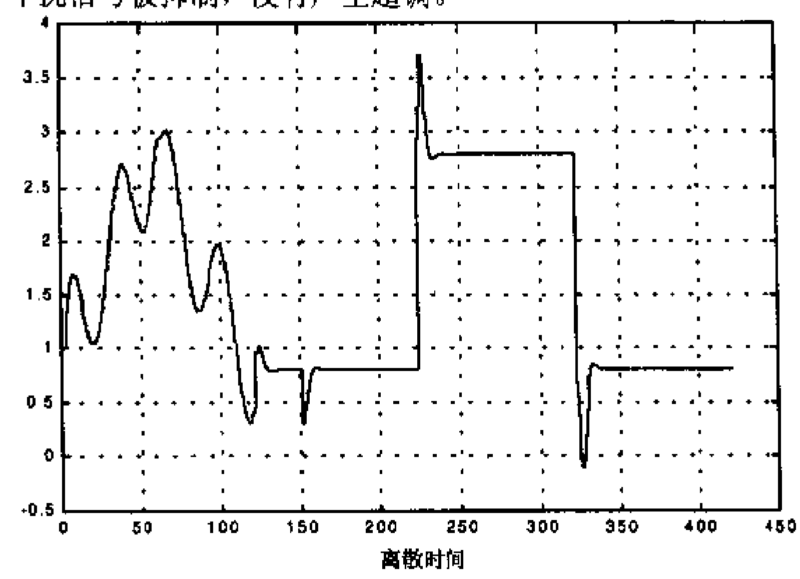


图 3-48　应用的控制信号

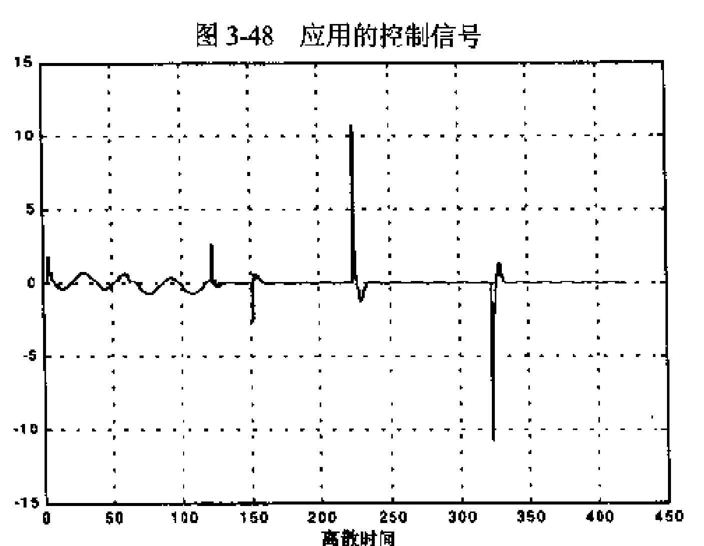


图 3-49　模糊调节器的输出量

图 3-50、图 3-51 为误差及其导数曲线。

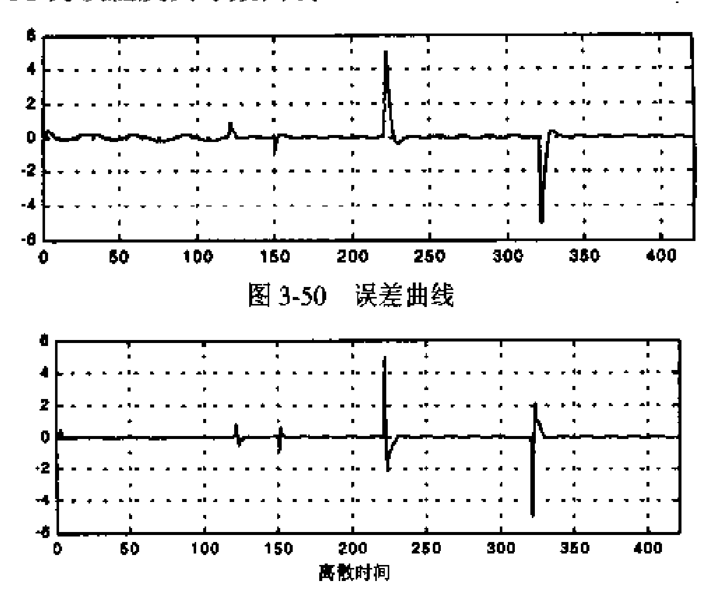


图 3-50　误差曲线

图 3-51　误差导数曲线

分析每一时刻的调节器输出，很容易得出模糊规则。

3.6.2　用工具箱命令实现模糊调节器

文件 prog_flou3.m 使用模糊逻辑工具箱的命令编制上节（通过图形界面）创建的模糊调节器。

prog_fuzzy3.m file

```
% Sugeno's type regulator
sys_fuzzy = newfis('reg_sug2','sugeno');

% error and its variation definition interval
interv_err = [-10 10] ; interv_derr = [-1 1];
sys_fuzzy = addvar(sys_fuzzy,'input','err',interv_err);

% input variables fuzzy sets definition
sigma = 8.5;
sys_fuzzy = addmf(sys_fuzzy,'input',1,'Neg','gaussmf',...
            [sigma min(interv_err)]);
sys_fuzzy = addmf(sys_fuzzy,'input',1,'Pos','gaussmf',...
            [sigma max(interv_err)]);

% adding the 2nd input variable :error derivative
sys_fuzzy = addvar(sys_fuzzy,'input','d_err',interv_derr);
sigma = 0.85;
sys_fuzzy = addmf(sys_fuzzy,'input',2,'N','gaussmf',...
            [sigma min(interv_derr)]);
sys_fuzzy = addmf(sys_fuzzy,'input',2,'P','gaussmf',...
            [sigma max(interv_derr)]);
% regultor's output variable definition
interv_cde = [-10 10];
```

```
sys_fuzzy = addvar(sys_fuzzy,'output','cde',interv_cde);

% the 4 fuzzy sets definition of the output variable
sys_fuzzy = addmf(sys_fuzzy,'output',1,'NN','linear',[2 1 0]);
sys_fuzzy = addmf(sys_fuzzy,'output',1,'PN','linear',[1 1 0]);
sys_fuzzy = addmf(sys_fuzzy,'output',1,'NP','linear',[1 1 0]);
sys_fuzzy = addmf(sys_fuzzy,'output',1,'PP','linear',[2 1 0]);

rules = ['if err is Neg and d_err is N then cde is NN';
        'if err is Neg and d_err is P then cde is NP';
        'if err is Pos and d_err is N then cde is PN';
        'if err is Pos and d_err is P then cde is PP'];

sys_fuzzy = parsrule(sys_fuzzy,rules, 'verbose');
```

选取一个常量作为输出量代替输入量的线性组合。所以，在输出量的定义区间选择一个足够大的数。下面研究修改输出量，考虑下列 4 个常量模糊集：

```
NG= 10, N= -5, P=5, et PG- 10
```

在文件 prog_flou4.m 中创建模糊调节器 reg_sug3.fis。下列代码用来修改文件 prog_flou3.m 中的调节器。

prog_fuzzy4.m(modification in prog_flou3.m file)

```
% 4 constants outputs singletons definition
sys_fuzzy = addmf(sys_fuzzy,'output',1,'PG','constant', 10 );
sys_fuzzy = addmf(sys_fuzzy,'output',1,'P',' constant',5 );
sys_fuzzy = addmf(sys_fuzzy,'output',1,'N',' constant',-5 );
sys_fuzzy = addmf(sys_fuzzy,'output',1,'NG',' constant ',-10 );

rules = ['if err is Neg and d_err is N then cde is NG';
        'if err is Neg and d_err is P then cde is N';
        'if err is Pos and d_err is N then cde is P';
        'if err is Pos and d_err is P then cde is PG'];
sys_fuzzy = parsrule(sys_fuzzy,rules, 'verbose');

% saving the fuzzy system to disk : reg_sug32 .fis
writefis (sys_fuzzy, 'reg_sug3' )
```

调节器特征（如类型、输入或输出特征、蕴含方法、去除离散化方法等）可由 Edit 指令以 ASCII 文本显示。

```
[System]
Name='reg_sug3'
Type='sugeno'
NumInputs=2
NumOutputs=1
NumRules=4
AndMethod='prod'
OrMethod='probor'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='wtaver'
```

```
[Input1]
Name='err'
Range=[-10 10]
NumMFs=2
MF1='Neg':'gaussmf',[8.5 -10]
MF2='Pos':'gaussmf',[8.5 10]

[Input2]
Name='d_err'
Range=[-1 1]
NumMFs=2
MF1='N':'gaussmf',[0.85 -1]
MF2='P':'gaussmf',[0.85 1]

[Output1]
Name='cde'
Range=[-10 10]
NumMFs=4
MF1='PG':'constant', 10
MF2='P': 'constant', 5
MF3='N': 'constant', -5
MF4='NG': 'constant', -10
[Rules]
1 1, 4 (1) : 1
1 2, 3 (1) : 1
2 1, 2 (1) : 1
2 2, 1 (1) : 1
```

默认时，Sugeno 模糊系统的 product 方法代替 AND，概率算子 OR 代替 OR，min 代替蕴含，max 代替规则合成，wtaver 方法去除模糊化。

概率算子 OR 或代数和如下式定义：

```
a Ou_prob b = a + b   a b
```

文件 cde_sug.m 中要用到 reg_sug3.fis 矩阵，通过 reg=readfis('reg_sug3')指令有如下结果（见图 3-52）。

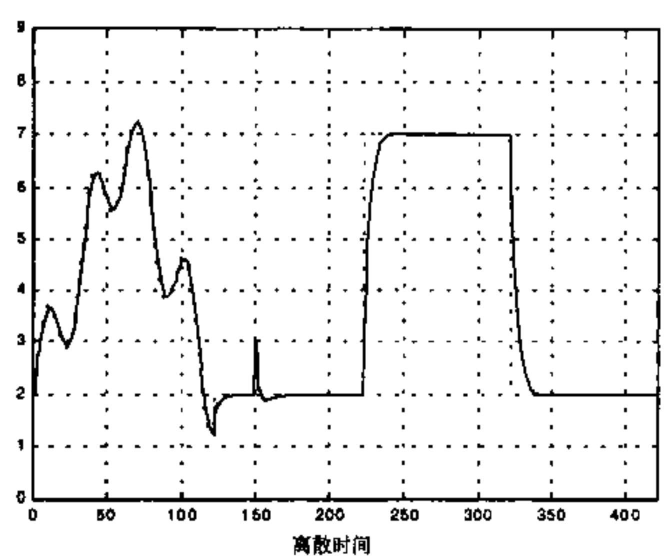


图 3-52　模糊调节器的目标信号和过程输出信号

增益等于 0.2。

t=150 的离散时刻，抗干扰的结果产生了一个弱的超调。

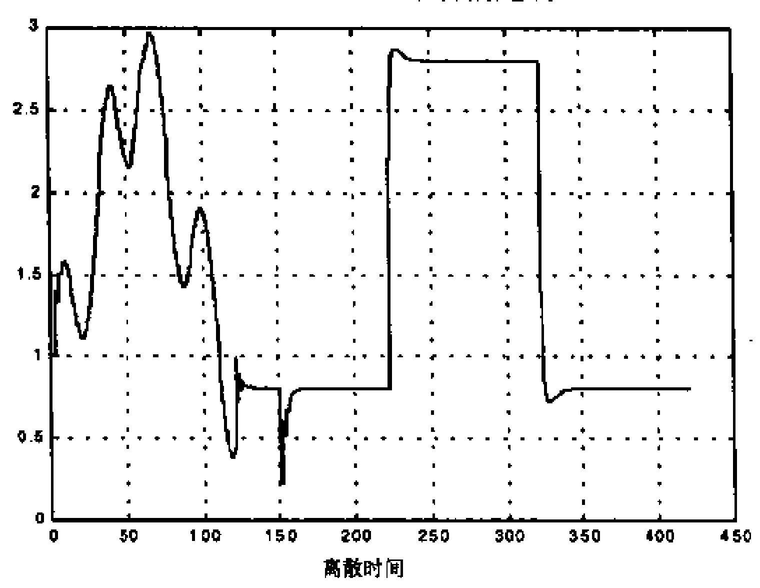


图 3-53　应用的控制信号

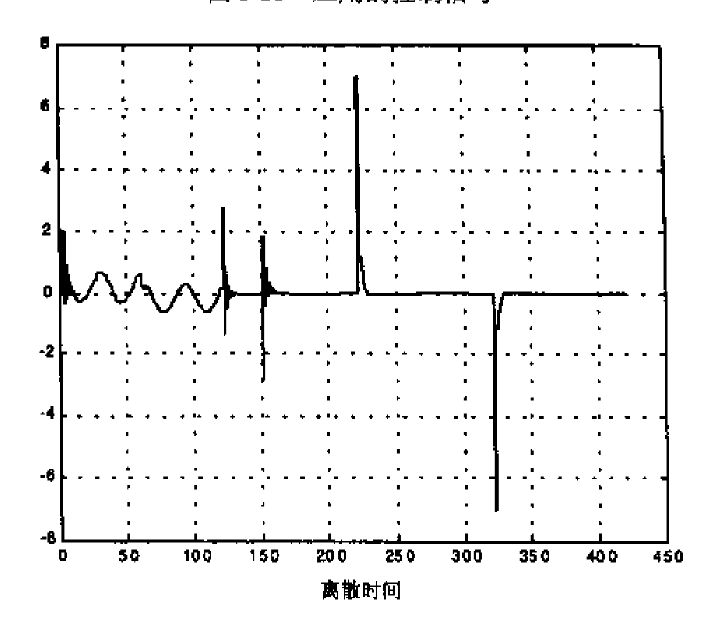


图 3-54　模糊调节器的输出量

在输入变量定义在同样的区间上时，如果输出为输入的线性组合，控制量较不稳定。由上图可看出在加入干扰和给定信号类型变化时，调节器输出有微小的振幅。为获得更稳定的控制，必须增加输入模糊集和输出常量的个数。

文件 prog_flou5.m 中，误差及其导数定义在[-10，10]区间，各有 3 个高斯隶属函数。

prog_flou5.m

```
% creation of Sugeno type regulator
sys_fuzzy = newfis('reg_sug4', 'sugeno');

% input variable and its variation definition
interv_err = [-10 10];
```

```
interv_derr = [-10 10];
sys_fuzzy = addvar(sys_fuzzy,'input','err',interv_err);

% fuzzy sets definition
% gaussian functions with standard deviation of 8.5 and means of -10, 0 et
10
sigma = 8.5;
sys_fuzzy = addmf(sys_fuzzy,'input',1,'Neg','gaussmf',...
            [sigma min(interv_err)]);
sys_fuzzy = addmf(sys_fuzzy,'input',1,'Nil','gaussmf',...
            [sigma mean(interv_err)]);
sys_fuzzy = addmf(sys_fuzzy,'input',1,'Pos','gaussmf',...
            [sigma max(interv_err)]);

% adding tne 2nd input variable
sys_fuzzy = addvar(sys_fuzzy,'input','d_err',interv_derr);

% gausian functions with standard deviation of 0.5 and means -10, 0 et 10
sys_fuzzy = addmf(sys_fuzzy,'input',2,'Negat','gaussmf',...
            [sigma min(interv_derr)]);
sys_fuzzy = addmf(sys_fuzzy,'input',2,'Zero','gaussmf',...
            [sigma mean(interv_derr)]);
sys_fuzzy = addmf(sys_fuzzy,'input',2,'Posit','gaussmf',...
            [sigma max(interv_derr)]);

% output regulator definition
interv_cde = [-10 10];
sys_fuzzy = addvar(sys_fuzzy,'output','cde',interv_cde);

% définition of th 5 output constant singletons
sys_fuzzy = addmf(sys_fuzzy,'output',1,'GN','constant',-10);
sys_fuzzy = addmf(sys_fuzzy,'output',1,'PN','constant', -5);
sys_fuzzy = addmf(sys_fuzzy,'output',1,'ZZ','constant',0);
sys_fuzzy = addmf(sys_fuzzy,'output',1,'PP','constant',5);
sys_fuzzy = addmf(sys_fuzzy,'output',1,'GP','constant',10);

rules= ['if err is Neg and d_err is Negat then cde is NG ';
       'if err is Neg and d_err is Zero then cde is PN ';
       'if err is Neg and d_err is Posit then cde is ZZ ';
       'if err is Nul and d_err is Negat then cde is PN ';
       'if err is Nul and d_err is Zero then cde is ZZ  ';
       'if err is Nul and d_err is Posit then cde is PP ';
       'if err is Pos and d_err is Negat then cde is ZZ ';
       'if err is Pos and d_err is Zero then cde is PP  ';
       'if err is Pos and d_err is Posit then cde is PG '];
   sys_fuzzy = parsrule(sys_fuzzy,rules,'verbose');

% saving fuzzy system on disk under the name reg_sug2.fis
writefis(sys_fuzzy,'reg_sug4')

% intermediate variables cancellation
clear sigma interv_cde regles base interv_derr base1 interv_err
```

应用了文件 cde_sug.m，其中调节器为 reg_sug4.fis。

增益等于 0.2 时，除了给定信号变化很大时，跟踪效果很好（见图 3-55）。

由于控制律中应用积分环节，无论增益值为多少，稳态误差都为零。

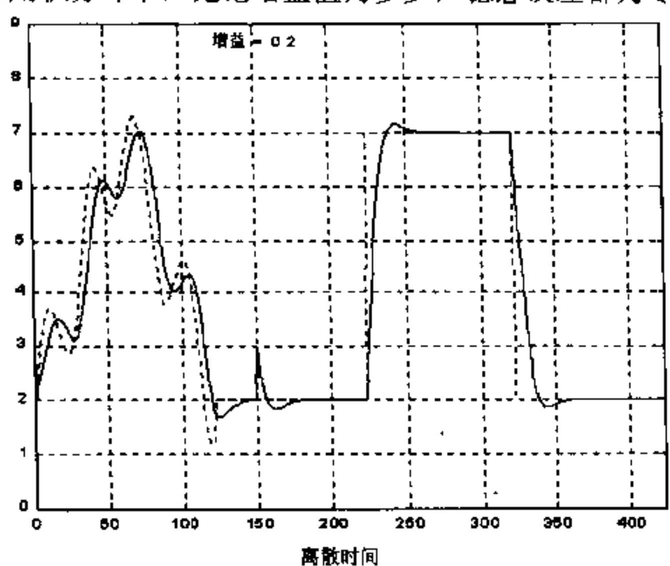


图 3-55　模糊调节器的目标信号和过程输出信号

增益等于 0.7 时，跟踪效果更好。如果给定信号是方波，增益值将影响响应时间（见图 3-56）。

文件 sugen_cde.m 显示了调节器输出增益对闭环系统时间响应的影响。

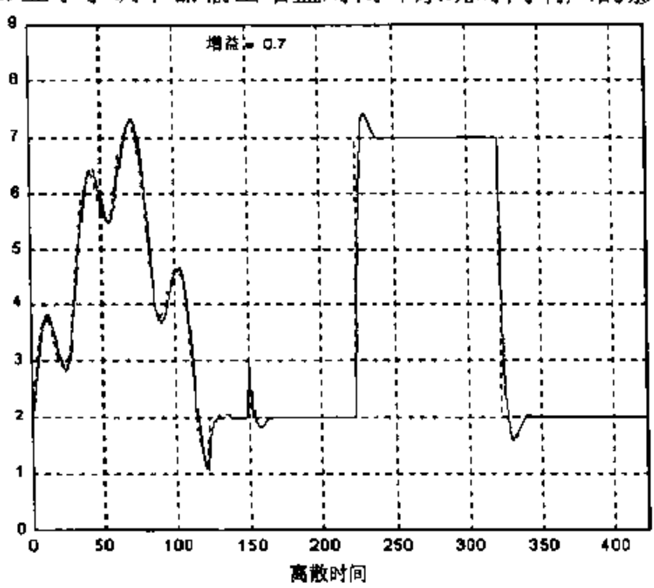


图 3-56　模糊调节器的目标信号和过程输出信号

sugen_cde.m file

```
% Régulateur flou de Sugeno
% effet du gain sur le temps de réponse du système bouclé

close all

% lecture du régulateur de Sugeno
```

```
reg = readfis('reg_sug4');
t = 0:600;

% signal de consigne carré

r = [zeros(1,50) 5*ones(1,100) zeros(1,50)];
r = [r r r]+2;

% différentes valeurs du gain multiplicatif
k = 0.1*(t<=200)+0.2*((t>200)&(t<=400))+0.9*((t>400)&(t<=600));

% control signal and error initialisation
u = ones(1,2); y = r;
err = zeros(1:2);
for i = 3:length(r)-1
   % reading of the process output
   y(i) = 0.8*y(i-1)+u(i-1)-0.5*u(i-2);
   % error and its derivative
   err(i) = r(i+1)-y(i);
   d_err(i) = err(i)-err(i-1);
   % control to be applied
   x = [err(i) d_err(i)];
   du(i) = evalfis(x,reg);
   u(i) = u(i-1)+k(i)*du(i);
end
% dispalying resullts
figure(1)
plot(r,':')
axis([0 length(r) 0 9]), hold on
plot(y), grid
title1 = 'fuzzy regulation';
title2 = 'target and process output signals';
title(strcat(title1,title2))
xlabel('discrete time')
ylabel('Sugeno''s method')
figure(2), stairs(u)
xlabel('discrete time'), grid
title('applied control signal')
figure(3), stairs(du)
title('regulator output')
xlabel('discrete time'), grid
```

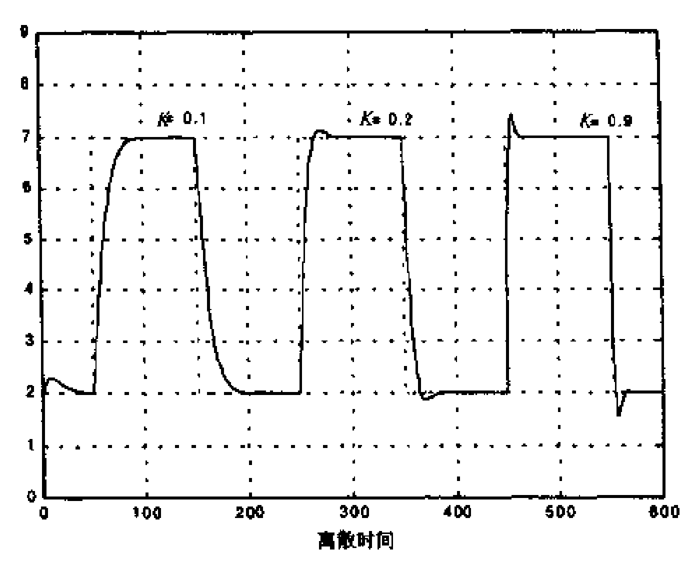


图 3-57 模糊调节器的目标信号和过程输出信号

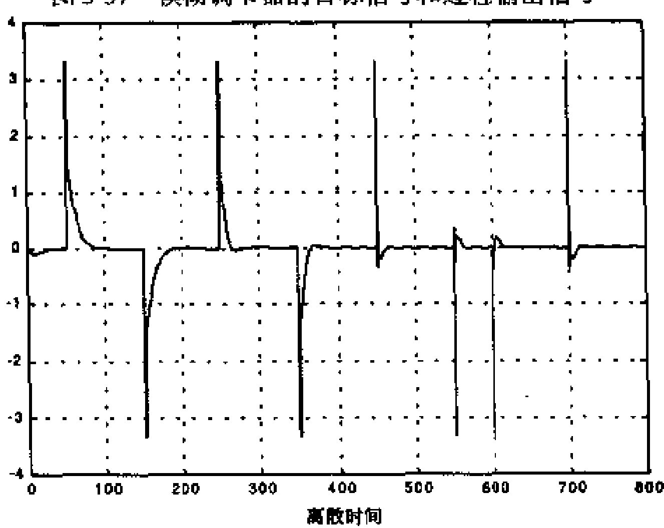


图 3-58 模糊调节器的输出量

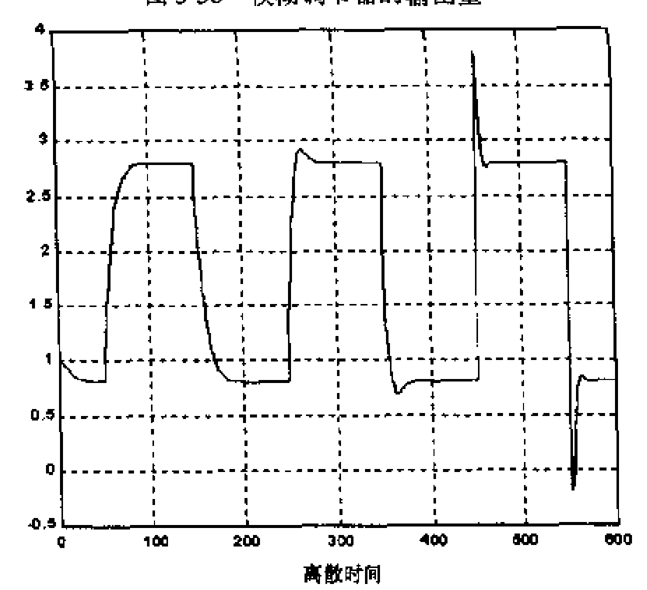


图 3-59 应用的控制信号

第4章 神 经 网 络

4.1 简介

神经细胞即神经元，是中央神经系统的基本单元。中央神经系统是由约 1 000 亿的神经细胞组成的。神经细胞通过树突接收神经冲动，经过整合产生一个新的神经冲动，然后通过轴突传给下一个神经细胞（见图 4-1）。

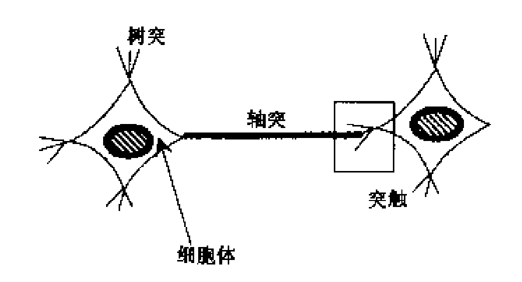


图 4-1 生物神经元

麦卡洛克（MacCulloch）与皮茨（Pitts）在 20 世纪 50 年代根据神经细胞的结构研制出了生物神经元模型。这种结构的神经元接收和发送的信号都是（0/1）的二进制数。输入信号与对应权值的乘积的总和与一个特定的阈值进行比较。如果超过阈值，则神经元自行激活；如果没有超过阈值，则神经元不传递任何信号（见图 4-2）。

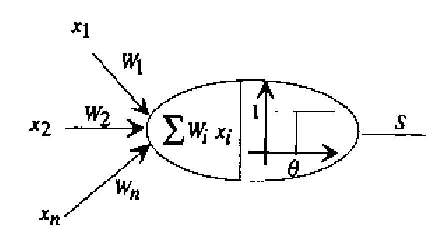


图 4-2 生物神经元模型

刺激物的输入信号与对应权值的乘积的总和使得神经元被激活，然后通过一个激活函数将信号送到一个输出端：

$$s = f(W_i x_i) = \begin{cases} 1 & si W_i x_i > \theta \\ 0 & si W_i x_i \le \theta \end{cases}$$

根据所处理的数据的类型(真值或二元的)以及所达到的功能的不同（如建模、模式识别、分类等），存在许多不同类型的神经网络。感知器神经网络被认为是分类领域里的第一个神经网络。

本书处理的应用问题只与信号处理有关，所以下面讨论线性自适应神经网络和含有隐层的神经网络。

4.2 线性自适应神经网络

4.2.1 结构

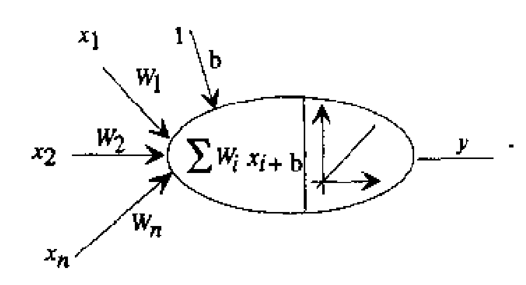


图 4-3 Adaline 型神经元结构

Adaline 型的神经元将接收到的输入信号与权值乘积的总和再加上一个偏移量。这个激活函数完成一个线性传递功能（见图 4-3）。

4.2.2 训练算法

神经网络的训练包括修改每一步的任何权值和偏移量，以逐步减小使用 Widrow-Hoff 规则的均方误差的和。每一步训练的输出误差是目标输出 t 和网络实际输出的差。第 k 步最小化的量是网络输出误差的方差：

$$E_k = e_k^{\mathrm{T}} e_k = (t_k - y_k)_k^{\mathrm{T}}(t_k - y_k) = \frac{1}{2}(t_k^{\mathrm{T}} t_k + y_k^{\mathrm{T}} y_k - 2y_k^{\mathrm{T}} t_k)$$

这个量的梯度和如下给出的权值矩阵一致：

$$\nabla E_{k/W} = \frac{1}{2}\nabla\left[y_k^{\mathrm{T}} y_k - 2y_k^{\mathrm{T}} t_k\right]_W$$

这个梯度可以按下式计算：

$$\nabla E_{k/W} = \frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial y_k}\frac{\partial y_k}{\partial W}$$

根据 E_k 的表达式和 $yk = Wx_k + b$，偏导数

$$\frac{\partial(Wx_k + b)}{\partial W} = x_k^{\mathrm{T}}$$

权值更新是由梯度下降法产生的，(k+l)步的权值矩阵为

$$W(k+1) = W(k) - \eta\nabla E_{k/W} = W(k) + \eta(t_k - y_k)x_k^{\mathrm{T}}$$

其中η为训练增益。所以得到偏移量修改的表达式

$$b(k+1) = b(k) - \eta\nabla E_{k/W} = b(k) + \eta(t_k - y_k)$$

总的来说，一个 Adaline 型的神经网络只有一个含有 S 个神经元节点的层。

如果输入向量是 R 维的，权值矩阵 W 是(S,R)维的。在每一步计算输出值 $y = W\,x + b$ 和目标向量，这些值在训练算法中要用到。

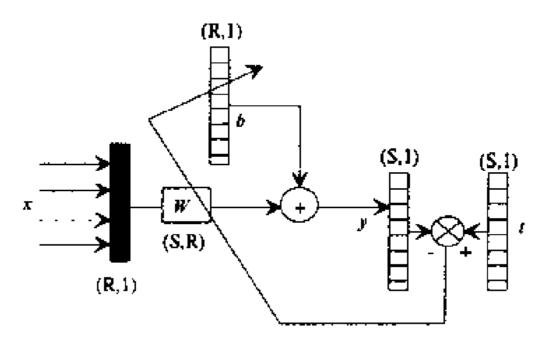


图 4-4　训练信号流图

“神经网络工具箱”提供了 learnwh 函数来实现这个算法。

```
[dw,db]= learnwh(x,e,eta)
```

- e　输出误差 $e=t-y$, (S,1) 维；
- x　输入向量，(R,1) 维；
- eta　训练增益；
- dw　权值的变化量, (S,R) 维；
- db　偏移量的变化量, (S,1) 维。

4.2.3　应用领域

4.2.3.1　过程识别

由 Adaline 型神经元构成的神经网络能够解决线性系统问题。通过指令

```
[w,b]=solvelin(x,t)
```

用代数方法计算出权值 w 和偏移量 b 就可以决定一个 Adaline 神经网络。

以一个 ARMA 模型的识别为例：

$$H(z)=z^{-1}\frac{b_1+b_2z^{-1}}{1-a_1z^{-1}}$$

其输入与输出方程可以写成如下形式：

$$y(t)=a_1y(t-1)+b_1u(t-1)+b_2u(t-2)$$

也可表示为：

$$y(t)=\begin{bmatrix}a_1 & b_1 & b_2\end{bmatrix}\begin{bmatrix}y(t-1)\\u(t-1)\\u(t-2)\end{bmatrix}$$

在采样时刻 $t=kT$ 时，输出 $y(t)$ 可以认为是一个线性神经元的目标输出，如图 4-5 所示。

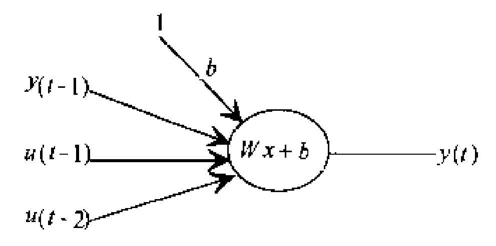


图 4-5　线性神经元结构

其中

$$x=[y(t-1)\quad u(t-1)\quad u(t-2)]^{\mathrm{T}}，b=0$$

文件 ada1_ident1.m 仿真了一个 ARMA 模型并决定神经元的输入和目标输出。

ada1_ident1.m

```
% identification by solvelin command
clear all, close all, clc

% model parameters to be identified
A = conv([1 -0.8],[1 0]);
B = [1 -0.5];

% PRBS input signal (length 1023)
u = (-1).^(1:10);
N = 100; % number of points
for i = 11:N
  u(i) = -u(i-7)*u(i-10);
end
y = (dlsim(B,A,u))';

% displaying the input-output signals
plot(u)
hold on
plot(y)
xlabel('samples')
title('input and output signals of the system to be identified')
% retarded signals
y1 = delaysig(y,1)
y1 = y1(2,:);

u1 = delaysig(u,2);
u2 = u1(3,:);
u1 = u1(2,:);

% suppression of the nil retarted signals elements
u1 = u1(3:N);
u2 = u2(3:N);
y1 = y1(3:N);
y = y(3:N);

% neuron input
x = [u1;u2;y1];
t = y;
% using the solvelin command
[w,b] = solvelin(x,t)
```

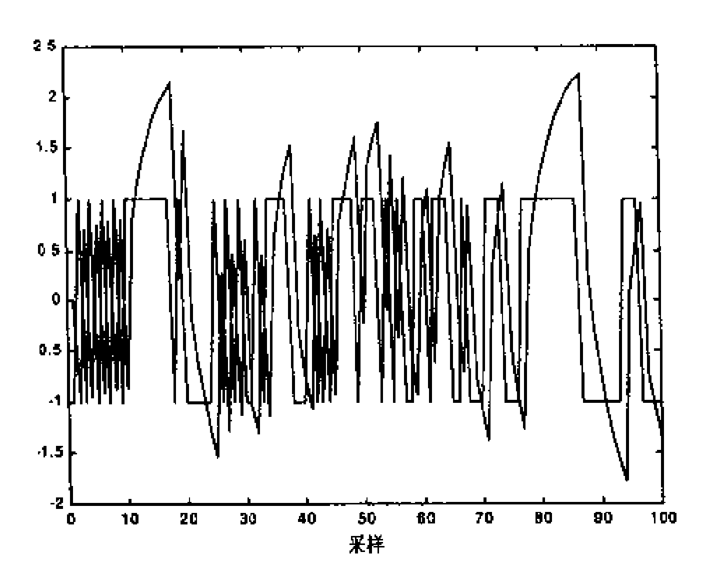


图 4-6　辨识系统的输入、输出信号

得到参数 a_1、b_1和 b_2 的精确值为这个线性神经元的权值矩阵 **W** 的元素。相对应于常数项的偏移量可以忽略。

```
W =
   1.0000    -0.5000     0.8000
b =
   -5.9217e-017
```

在一些情况下，这个指令用矩阵计算（如求逆）不会产生结果。我们必须用 learnth 指令编程实现 Widrow-Hoff 训练规则。

adal_ident2.m

```
% Adaline neuron, Widrow-Hoff law
clear all, close all, clc
% model parameters to be identified
A = conv([1 -0.8],[1 0]); B = [1 -0.5];
% PRBS input signal (length 1023)
u = (-1).^(1:10);
N = 500; % number of points
for i = 11:N
  u(i) = -u(i-7)*u(i-10);
end
y = (dlsim(B,A,u))' ;

% weights and bias initialisation
[W,b] = rands(1,3);
weights_W = cat(3,W);
bias_b = cat(3,b);

eta = 0.1;    % training gain

  for i = 3:length(u)-1

     % 1st stimulus
     p = [y(i-1) u(i-1) u(i-2)]';
```

```
    % output neuron
    o = W*p + b;

    % error
    e = y(i) - o;

    % weight and bias variations
    [dw,db] = learnwh(p,e,eta);

    % updating the W weights and b bias
    W = W + dw;
    b = b + db;

    % saving weights and bias
    weights_W = cat(3,weights_W,W);
    bias_b = cat(3,bias_b,b);

  end

disp('final values of weights and bias');
W, b

% plotting the weights evolution
for k = 1:3
   x = weights_W(1,k,:);
   plot(x(:)), hold on
end
eta = num2str(eta);
title(['W weights and b bias evolution, \eta = ' eta])
xlabel('samples')
plot(bias_b(:),':')
```

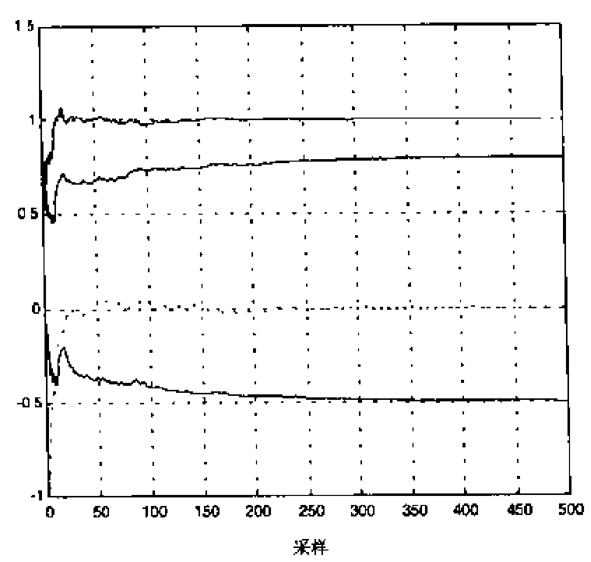


图 4-7　权值 W 和偏移量 b（η=0.1）的变化曲线

经过训练得权值和偏移量如下：

```
final values of weights and bias

W =
    0.7979     0.9999     - 0.4981
b =
  -9.1189e -004
```

4.2.3.2 信号预测

不用神经网络工具箱命令编程实现 Widrow-Hoff 训练算法。

pred_adal.m

```
% One step ahead signal prediction
clear all; close all
clc

time = 0:0.05:10;
r = 0.5 * sin(time*2*pi);

% random initialisation of the weights and bias
S = 1;  % Adaline neuron
R = 5;  % inputs number
[W,b] = rands(S,R);

% weights and bias initial values
weights_W = cat(3,W);
bias_b = cat(3,b);
eta = 0.1;        % training gain

   for i = 5:length(r)-1
   % 1st stimulus
   x = [r(i) r(i-1) r(i-2) r(i-3) r(i-4)]' ;
   % output
   y = W*x+b;
   dr_est(i+1) = y;
   r_est(i+1) = dr_est(i+1)+r(i);

   % error
   e = r(i)-r(i-1)-dr_est(i);

   % updating of the weights and bias
   W = W+eta*e*x'; b = b+eta*e;

   % saving the weights and bias
   weights_W = cat(3,weights_W,W);
   bias_b = cat(3,bias_b,b);
end

disp('weights and bias final values values :'); W, b

% plotting different signals
```

```
figure(1), hold on
plot(time, r_est), plot(time,r,':')
title(['real and predict signals  \eta = ' num2str(eta)])
xlabel('time')

figure(2), plot(e)
title(['prediction error,  \eta = ' num2str(eta)])
hold on, plot(time,r-r_est)
xlabel('time')

hold off

% weights evolution curve
figure(3)
for k = 1:R
   x = weights_W(1,k,:);
   plot(x(:))
   hold on
end
xlabel('samples')
hold off
title(['W weights evolution curve,  \eta = ' num2str(eta)])

% plotting bias evolution curve
figure(4), plot(bias_b(:))
title(['bias evolution curve,  \eta = ' num2str(eta)])
xlabel('samples')
```

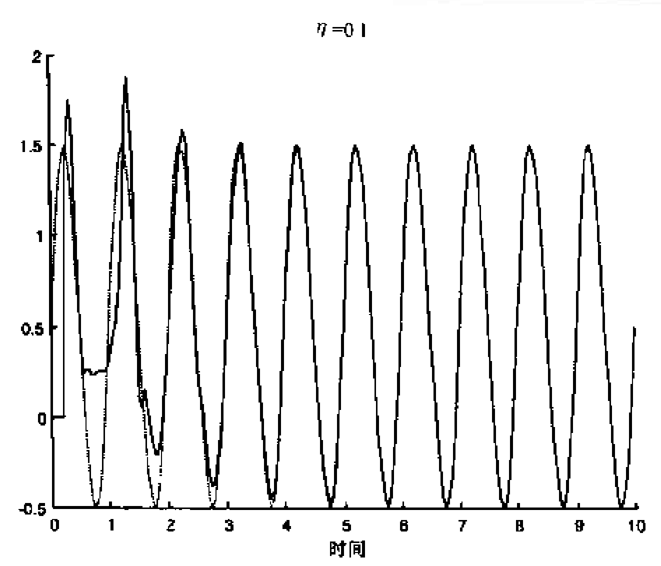


图 4-8　实际信号和预测信号

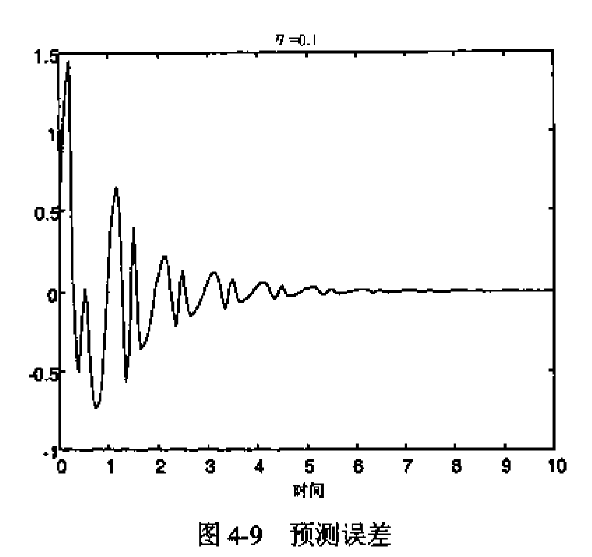


图 4-9　预测误差

```
weights and bias final values :
W =
   0.3424   - 0.7085    0.6750    -0.3263    -0.2341
b =
   0.1254
```

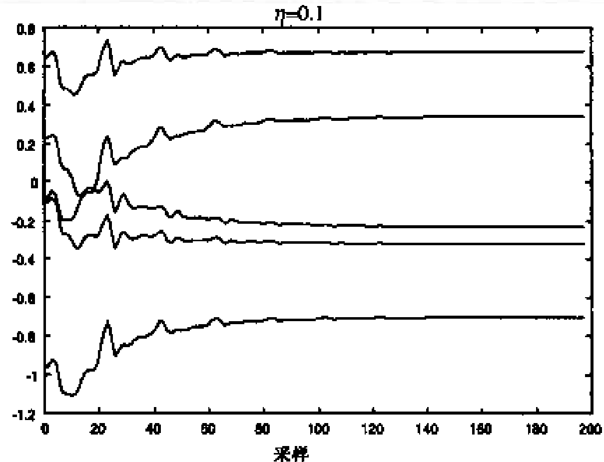


图 4-10　权值 W 变化曲线

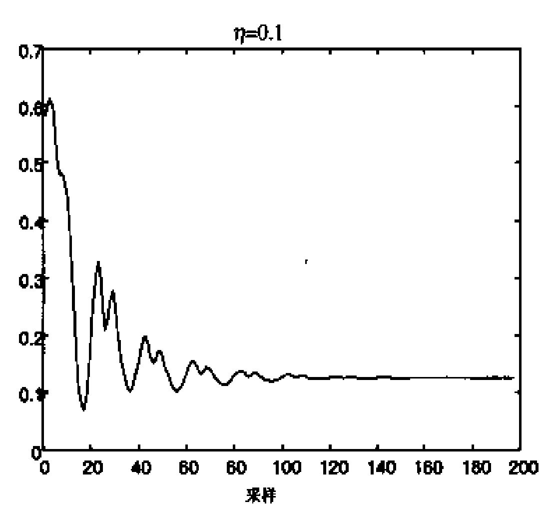


图 4-11　偏移量变化曲线

4.2.3.3 消除干扰

在工业中，有用信号 x 并不能单独传输，其中往往有引入干扰的寄生信号 p。实际传输信号由有用信号 x 和与 p 成比例的噪声构成（见图 4-12）。

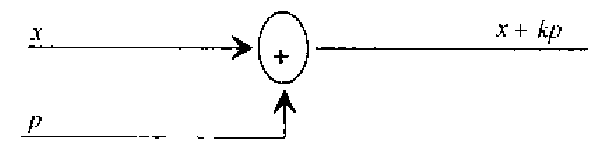


图 4-12 系统结构

用神经网络消除干扰的原理是把 p 作为线性神经元的输入重构感染噪声的信号。由于神经元是线性的，因此它输出的一部分与噪声 p 成比例，这就是干扰。神经元输出与目标输出之间的误差即为要提取的有用信号 x（见图 4-13）。

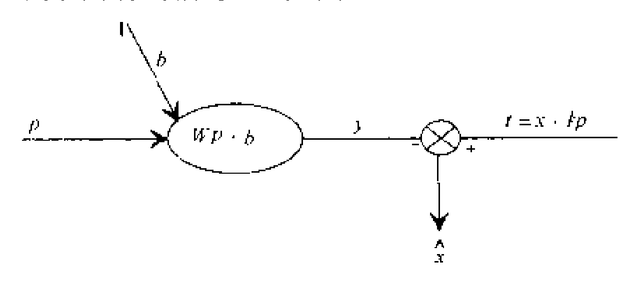


图 4-13 线性神经元系统结构

例如，在一个噪声较大的环境中打电话，如车间、机场等，想要传输最接近于有用信号 x 的信号。这时已知整个信号，可以用如下自适应函数：

```
[y, e, w, b] = adaptwh(W, b, x, t, eta)
```

- [y，e，w，b]　：输出、误差、最后的权值和偏移量；
- W，b，x，t，eta ：初始权值、偏移量、输入矩阵、目标和训练增益。

sup_interfl.m

```
% Interference cancellation
clear all, close all, clc
time = 0:0.1:10;
r = sin(time*4*pi);

% Random initialisation of the W weight and b bias
R = length(time);   % number of inputs
S = R;
% p parasite signal
p = randn(size(r));

% noised signal
t = r + 0.833*p;

% W and b initialisation
[W,b] = initlin(p,t);

figure(1), plot(time,t)
```

```
title('target to be predicted = noised signal'), xlabel('time')

[y,e] = adaptwh(W,b,p,t,0.1);

figure(2)
plot(r,':')
hold on
plot(e)
title('useful signal = error signal')
hold off

figure(3)
plot(r-e)
title('error = useful signal - reconstructed signal')
xlabel('time')

% final weight and bias
w, b
```

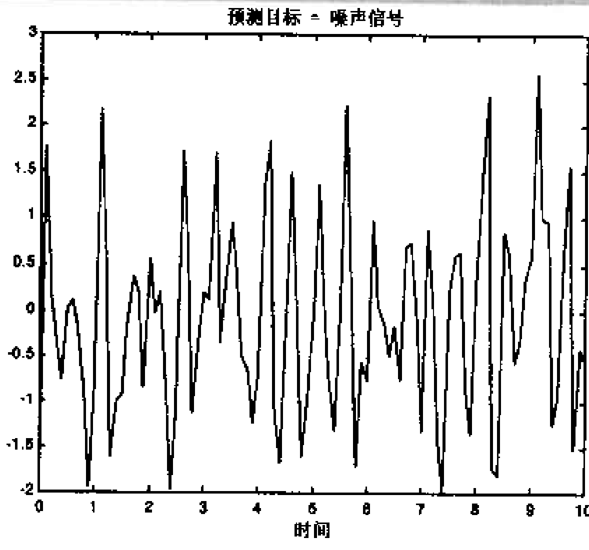


图 4-14　含有噪声的信号

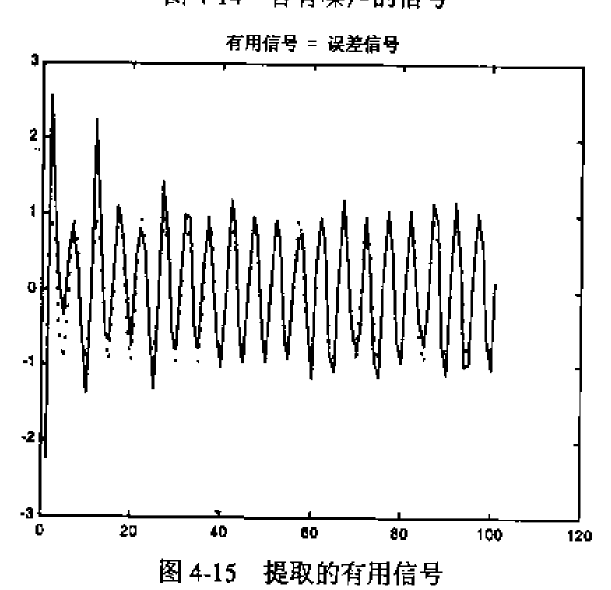


图 4-15　提取的有用信号

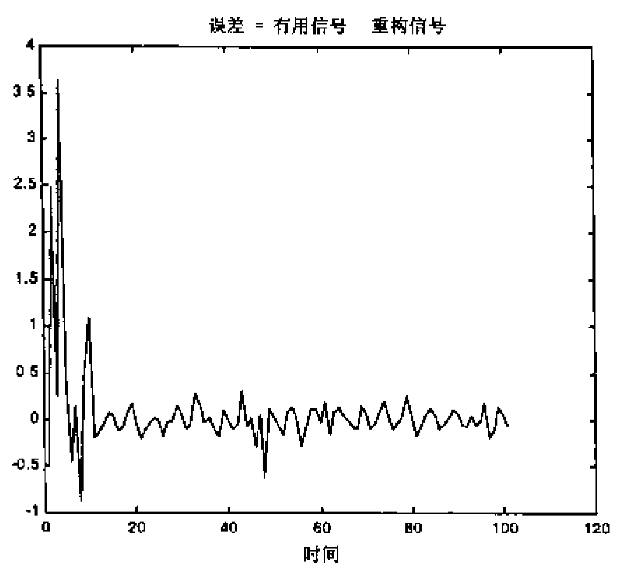


图 4-16 误差变化曲线

得到最后的权值和偏移量如下：

```
w =
   -0.9695
b =
   0.4936
```

在文件 sup_interf2.m 应用了“神经网络工具箱”函数 learnwh。

sup_interf2.m

```
% Interference cancellation
clear all, close all, clc
time = 0:0.05:20;
r = sin(time*pi);
% Random initialisation of the W weights and b bias
S = 1;
R = S;
% p parasite signal
p = randn(size(time));
% noised signal
t = r + 0.833*p;
% W and b initialisation
W = randn(1,1);
b = randn(1,1);
weight_W = W;
bias_b = b;

figure(1), plot(time,t)
title('target to be predicted = noised signal')
xlabel('time')
eta = 0.02;
for i = 1:length(time)
   % neuron's output
   y(i) = W*p(i)+b;

   % output error
   e(i) = t(i) - y(i);
```

```
    [dW,db] = learnwh(p(i),e(i),eta);
    % updating the weight and bias

    W = W+dW;
    b = b+db;

    % saving weight and bias
    weight_W = [weight_W,W];
    bias_b = [bias_b,b];
end

figure(2)
plot(time,r,':'), hold on, plot(time,e)
title('useful signal = error signal')
hold off
figure(3)
plot(time,r-e)
title('error = useful signal - reconstructed signal')
plot(time,zeros(size(t)))
xlabel('time')
```

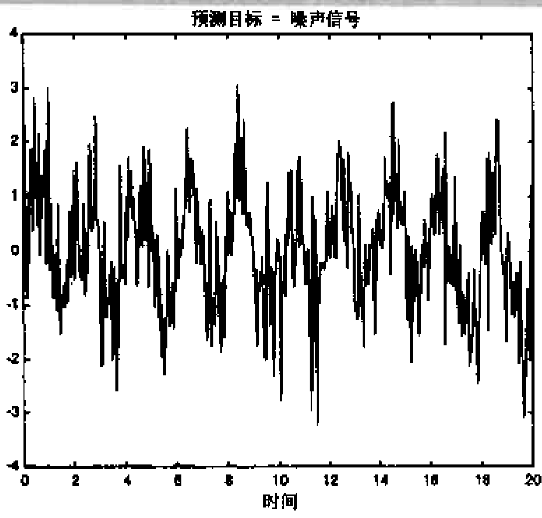


图 4-17　含有噪声的信号

图 4-18　提取的有用信号

```
>> w
w =  0.9466
>> b
b =  -0.2157
```

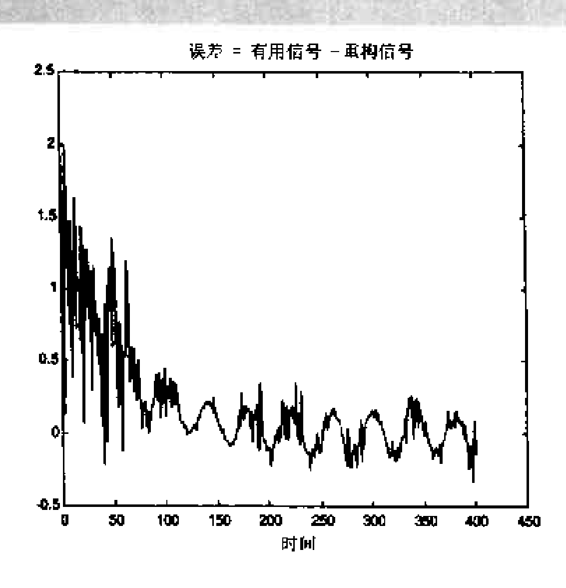


图 4-19　误差变化曲线

4.3　含有隐层的神经网络，误差反向传播

4.3.1　原理

这种类型的网络有一个或多个中间层，也就是所谓的隐层（见图 4-20）。

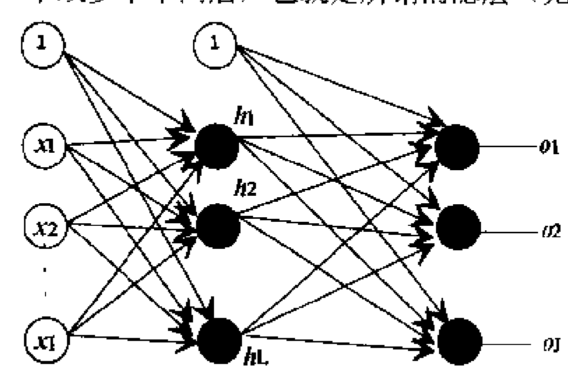


图 4-20　含有隐层的网络结构

输入层的节点通过一些连接向隐层节点传递刺激信息。隐层的输入是一个具有如下矩阵形式的向量：

$$h_in = Wx + w0$$

这里 W 和 $w0$ 分别代表(L, I)维的权值矩阵和(I, 1)维的偏移向量，x 是(J, 1)维的刺激信息向量。每一个隐层节点都通过一个传递函数将输入信息转化为相应的反应输出：

$$h = f(h_in) = f(Wx + w0)$$

同理，每一个输出层的神经元节点也通过一个传递函数将输入信息转化为相应的反应

输出。下面是输出层的节点反应输出：

$$o = g(Zh + z0)$$

用随机数初始化权值矩阵 W、Z 和偏移向量 $w0$、$z0$。这个神经网络的输出向量将与期望的目标向量进行比较。然后，计算出神经网络的输出误差。这个误差在神经网络中进行反向传播，目的是通过网络将误差信号沿原来的连接通路反传回去，并根据使均方误差最小的算法来修改权值矩阵和偏移向量的值。

4.3.2 传递函数

传递函数或激活函数是非线性递增的，允许引入一个阈值或饱和值。在 BP 算法中，需要激活函数的导数参与。因此，我们对那些容易计算出导数的函数非常感兴趣。当前用得最多的函数是单极性 S 函数、双极性 S 函数和双曲正切函数，它们可以把输入从区间[-∞，∞]分别变换成[0，1] 和 [-1，1]区间的输出。

- 单极性 S 函数

$$f_1(x) = \frac{1}{1+\mathrm{e}^{-x}}$$

$$f_1(x) = f_1(x)[1 - f_1(x)]$$

- 双极性 S 函数

$$f_2(x) = \frac{1-\mathrm{e}^{-x}}{1+\mathrm{e}^{-x}} = 2[f_1(x) - 0.5]$$

$$f_2(x) = \frac{1}{2}[1 + f_2(x)][1 - f_2(x)]$$

- 双曲正切函数

$$f_3(x) = \frac{\mathrm{e}^{x} - \mathrm{e}^{-x}}{\mathrm{e}^{x} + \mathrm{e}^{-x}}$$

$$f_3'(x) = \frac{4}{(\mathrm{e}^{x} + \mathrm{e}^{-x})^2}$$

文件 **f_activ.m** 给出了经常使用的激活函数。

f_activ.m file

```
close all
x = -6:0.1:6;
% unipolar sigmoid
sigm_unip = logsig(x);

% hyperbolic tangent
tnh = tanh(x);

% bipolar sigmoid
sigm_bip = 2*logsig(x)-1;

plot(x,tnh)
grid
gtext('hyperbolic tangent')
hold
```

```
plot(x,sigm_unip,':')
gtext('unipolar sigmoid')
plot(x,sigm_bip,'--')
gtext('bipolar sigmoid')
title('several activation functions');
```

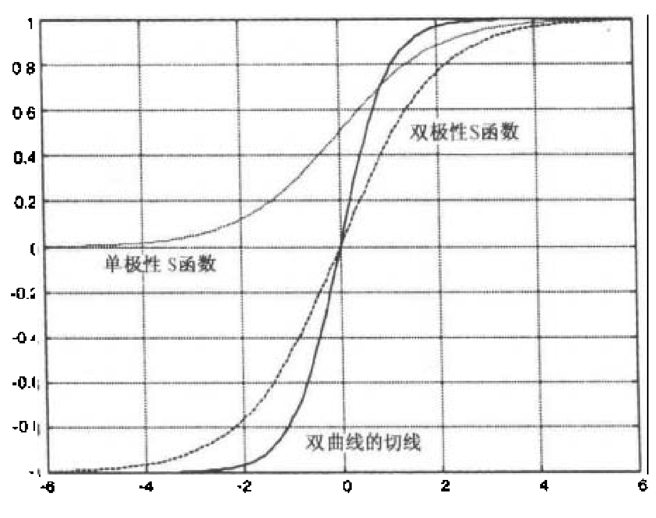


图 4-21 几种激活函数

这些函数的使用分别意味着神经网络输入和输出幅度的不同条件要求。因此，我们定义一个能够修改倾斜度和极限值的函数。如果只想修改倾斜度，我们考虑下面的函数：

$$f_4(x) = f_1(\sigma x) = \frac{1}{1+\mathrm{e}^{-\sigma x}}$$

它的导数为：

$$f_4'(x) = \sigma\, f_1(\sigma\, x)[1 - f_1(\sigma\, x)]$$

如果还要修改极限值，我们定义下面的函数：

$$g(x) = \alpha f_4(\sigma x) - \beta$$

这使得 $g(x_{\to\infty}) = \alpha - \beta$ 且 $g(x \to -\infty) = -\beta$。如果要让函数值对称于 $\pm\beta$，选择 $\alpha = 2\beta$。至于其他函数的导数，可用传递函数表达式来表示：

$$g'(x) = \frac{\sigma}{\alpha}[\beta + g(x)][\alpha - \beta - g(x)]$$

文件 sigm–var.m 实现了 S 函数。

sigm_Var.m

```
function [g, gp] = sig_param(x,par)
g = par(1)*logsig(par(3)*x)-par(2);
gp = (par(2)+g).*(par(1)-par(2)-g)*par(3)/par(1);
```

运行文件 sigm–var.m 画出了几个不同版本的 S 函数和它们的导数。

sigm_var.m

```
close all
x = -10:0.1:10;
% a_b_s : vector comprising alpha, beta and sigma parameters
abs1 = [4 2 2];

[g1,gp1] = sigm_param(x,abs1);
```

```
plot(x,g1)
text(-0.9,-1.5,'\leftarrow \alpha=4, \beta = 2, \sigma = 2',...
    'HorizontalAlignment','left')
hold on
abs2 = [4 2 1];
[g2,gp2] = sigm_param(x,abs2);
plot(x,g2,':')
text(-2,-1.5,'\alpha = 4, \beta= 2, \sigma= 1 \rightarrow ',...
   'HorizontalAlignment','right')

abs3 = [3 1 0.5];
[g3,gp3] = sigm_param(x,abs3);
plot(x,g3,'-.')
text(-1.6,0,'\alpha=3, \beta=1, \sigma = 0.5 \rightarrow ',...
    'HorizontalAlignment','right')

axis([-10 10 -2.5 2.5])
title('sigmoid with variable slope and extreme values')
xlabel('activation x')
gtext('g(x) = \alpha/(1+e^(-\sigma x)) - \beta')

% plotting the sigmoid's derivatives
figure(2)
plot(x,gp1)
hold on
plot(x,gp2,':')
plot(x,gp3,'-.')
title('sigmoid''s derivatives')
xlabel('activation x')
axis([-10 10 0 2.4])
```

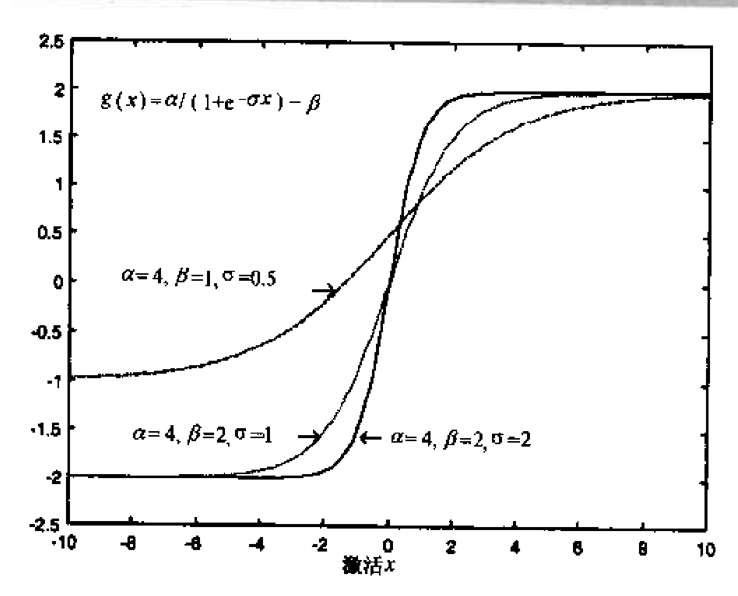


图 4-22　几种不同坡度和极值的 S 函数曲线

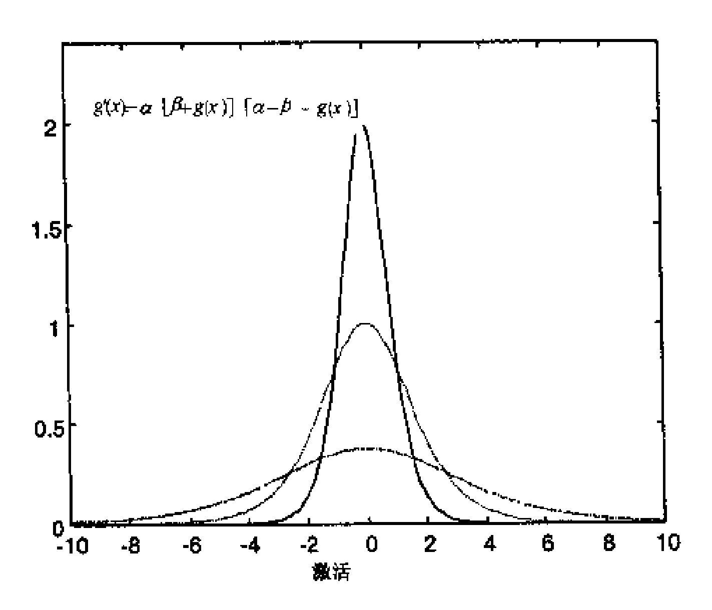


图 4-23　几种 S 函数的导数曲线

“神经网络工具箱”提供了下列传递函数：

- logsig　单极性 S 函数；
- tansig　双曲正切函数；
- purelin　线性传递函数；

以上每个函数的导数分别通过 deltalog、deltatan 和 deltalin 来计算。

- deltalog (o)　返回导数向量 o.*(l-o)。

4.3.3　BP 算法

我们记

- X　(I, N) 维刺激信息输入矩阵，N 代表神经网络的刺激信息的输入数目；
- x_k　刺激信息 n°k（矩阵 X 的 n°k 列）；
- T　(I, N)维的目标矩阵或者是理论输出；
- t_k　目标输出 n°k（矩阵 T 的 n°k 列）；
- h_k　(L,l)维的隐层节点输出；
- o_k　神经网络的反应输出, J 维的列向量；
- W　输入层节点和隐层节点之间的的连接权值矩阵 (L, I)维；
- $w0$　隐层节点单元的偏移量的向量 (L, 1)维；
- Z　隐层节点和输出层节点之间的的连接权值矩阵 (J, L)维；
- $z0$　输出层节点的偏移向量 (J, 1)维。

在每一个刺激信息 n° k 输入时，都要执行下面两个连续的阶段：

1）刺激信息在神经网络上的正向传递；

2）把从神经网络上得到的误差反向传播，目的是为了更新权值和偏移量的值，以便减少网络输出和目标输出之间的误差。

◆ 刺激信息在神经网络上的正向传递

每一个隐层节点计算它所有输入的总和，并且促使这些获取的信号通过传递函数进行传递。在输出层节点上也是这样进行信号传递的。输出向量 n° k 的矩阵形式如下式所示：

$$o_k = g[Z(f(Wx_k + w0) + z0]$$

其中 f 和 g 这两个传递函数分别代表了隐层和输出层的传递函数。将这个输出向量与期望输出向量 t_k 进行比较。误差按下式推导：

$$e_k = t_k - o_k$$

◆ 误差反向传播

神经网络的训练是由每一阶段对权值和偏移量的修改组成的，目的是最小化输出误差的平方和。BP 算法是基于梯度技术的。每次训练时最小化的量是神经网络输出误差的方差：

$$E_k = e_k^{\mathrm{T}} e_k = \frac{1}{2}\left(t_k^{\mathrm{T}} t_k + o_k^{\mathrm{T}} o_k - 2o_k^{\mathrm{T}} t_k\right)$$

$$\nabla E_{k/Z} = \frac{1}{2}\nabla[o_k^{\mathrm{T}} o_k - 2o_k^{\mathrm{T}} t_k]_{/Z}$$

如果考虑隐层和输出层的两个传递函数是一样的，那么神经网络的输出计算如下：

$$o_k = f(Zh_k + z0]$$

E_k 相对于 Z 矩阵的梯度计算如下：

$$\nabla E_{k/Z} = \frac{\partial E_k}{\partial Z} = \frac{\partial E_k}{\partial o_k}\frac{\partial o_k}{\partial(Zh_k + z0)}\frac{\partial(Zh_k + z0)}{\partial Z}$$

根据 E_k 的表达式，第 1 个偏导数如下：

$$\frac{\partial E_k}{\partial o_k} = o_k - t_k$$

第 2 个偏导数依赖于选择的传递函数的类型。在单极性的 S 函数的情况下，第 2 个导数如下表达：

$$\frac{\partial o_k}{\partial(Zh_k + z0)} - f(Zh_k + z0) \cdot *[1 - f(Zh_k + z0)] = o_k \cdot *(1 - o_k)$$

1 代表和输出向量 o_k 维数相同的单位矩阵，“ .* ”操作代表项对项的乘积或者代表笛卡尔乘积。

第 3 个偏导数计算如下：

$$\frac{\partial(Zh_k + z0)}{\partial Z} = h_k^{\mathrm{T}}$$

根据梯度下降法更新权值，(k+1)步的权值矩阵 Z 如下：

$$Z(k+1) = Z(k) - \eta\nabla E_{k/Z} = Z(k) + \eta(t_k - o_k) \cdot *o_k \cdot *(1 - o_k)h_k^{\mathrm{T}} = Z(k) + \eta\partial s\, h_k^{\mathrm{T}}$$

其中 η 为训练增益。

更新偏移向量 z0 方法同上。鉴于也是通过偏移量传递输入向量，易得：

$$z0(k+1) = z0(k) - \eta\nabla E_{k/z0} = z0(k) + \eta(t_k - o_k) \cdot *o_k \cdot *(1 - o_k) = z0(k) + \eta\partial s$$

输出误差 ∂_s 通过 Z 矩阵的转置矩阵反向传播到隐层的输出。

根据上面的 Z 矩阵更新表达式，其中 h_k 为隐层节点的输出向量，我们得到：

$$W(k+1)=W(k)-\eta\nabla E_{k/W}=W(k)+\eta Z(k)^{\mathrm{T}}\cdot *h_k\cdot *(1-h_k)x_k^{\mathrm{T}}=W(k)+\eta\partial_h x^{\mathrm{T}}$$

和

$$w0(k+1)=w0(k)-\eta\nabla E_{k/w0}=w0(k)+\eta Z(k)^{\mathrm{T}}\cdot *h_k\cdot *(1-h_k)=w0(k)+\partial_h \text{。}$$

“神经网络工具箱”提供了计算任何网络层的输出误差的函数。如果我们考虑先前定义的带一个隐层的神经网络，这些函数为 delta 函数，它们可以计算 ∂_s 和 ∂_h 的误差：

- deltalog (o) 如果 o 为这一层的输出向量，计算一个输出层误差的导数向量为 o.* (1 - o);
- deltalog (o, e) 计算 e.*o.* (1-o)。在上面的积分表达式里为 ∂_s；
- deltalog (h, ds, Z) 计算 $Z(k)^{\mathrm{T}}ds.*h*.*(1-h*)$。在上面的积分表达式里为 ∂_h。

这些针对单极性的 S 传递函数的不同句法与传递函数 deltatan 和 deltalin 的句法是一样的。

4.4 逆模式神经网络控制

逆模式神经网络过程是一个能够根据先前的输入输出值来计算离散时刻 $t=kT$ 时的输入（控制信号）$u(t)$的值的神经网络。

建模阶段需要过程的输入输出数据文件。对不同的过程模式，将应用一个富含频率的信号、一个频率在期望输出附近的伪随机二元序列信号(PRBS)。如果得到 N 组数据，即获得了一个 N 行 2 列的矩阵文件，2 列分别代表离散输入 $u(t)$和离散输出 $y(t)$。

考虑下面的传递函数：

$$H(z)=\frac{z^{-1}(0.3+0.05z^{-1})}{1-0.7z^{-1}}$$

在等于 5 的一个常量信号上叠加一个单位高度和 1023 长度的伪随机二元序列信号（PRBS）作为上述模型的控制信号，其中常量信号相应于稳态过程输出。从包含 $u(t)$和 $y(t)$的文件中计算出变量 $\Delta u(t)=u(t)-u(t-1)$和 $\Delta y(t)=y(t)-y(t-1)$。当使用单级性的 S 函数，我们必须标准化不同的信号 $u(t)$、$y(t)$、$\Delta u(t)$ 和 $\Delta y(t)$，使它们的值在 0.1~0.9 之间。

4.4.1 第一层网络结构

下面的神经网络提供每一个采样时刻的控制信号变量 $\Delta u(t)=u(t)-u(t-1)$。通过预测输出变量 $\Delta y(t+1)$得到一个一步预测控制（见图 4-24）。

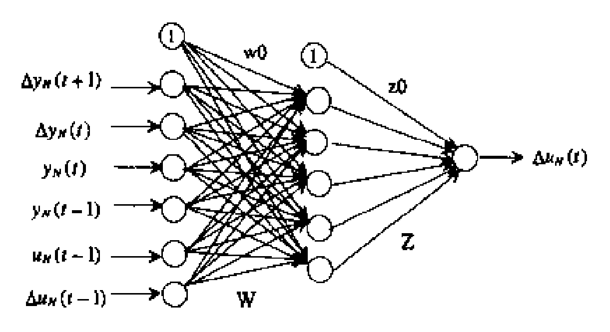


图 4-24 第一层网络结构

文件 model.m 能够得到和显示过程的输入和输出。

model.m

```
% inputs-outputs generation file of a process model
clear all; close all

% cosntant signal+ PRBS sequence
time = 200;
u = 5*ones(1,time);

% PRBS sequence generation
prbs = (-1).^(1:10);

for i = 11:length(u)
   prbs(i) = -prbs(i-10)*prbs(i-7);
end

u = u+prbs;

% model output
num = [0.3 0.05];
den = conv([1 0],[1 -0.7]);
y = (dlsim(num,den,u))';

% transient values suppression
u(1:10) = []; y(1:10) = [];

% displaying the control signal and the model output
stairs(u)
hold on
h = plot(y);
set(h,'LineWidth',2)
title('Control signal and model output')
xlabel('discrete time'), grid, hold off
axis([0 length(u) 3.5 7])
% intermediate variables suppression
clear num den h i prbs time
```

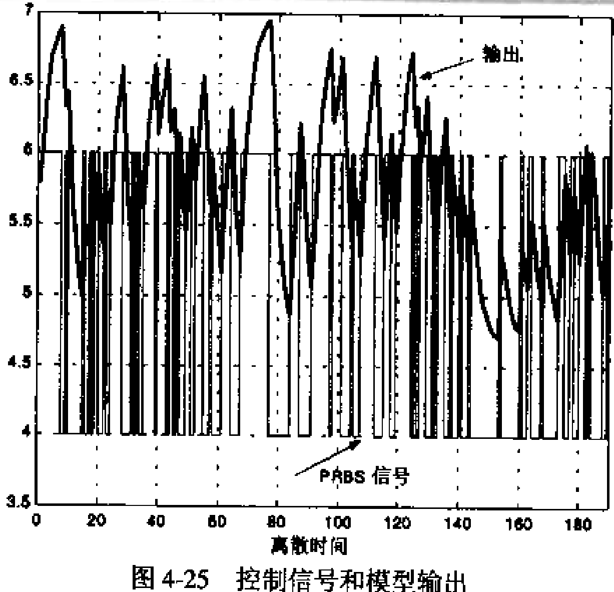


图 4-25　控制信号和模型输出

由前面的神经网络结构可知，必须标准化输入和输出信号以及它们的变化量。文件 es_norml.m 把这些不同的信号标准化，并且把它们保存在 es_N1.mat 中。在控制律里用到的标准化系数将被保存在 *ab_norml.mat* 文件里。

io-norml.m

```
% input and output variations
du = diff(u); dy = diff(y);

% suppression of the last u and y values
u(length(u)) = []; y(length(y)) = [];

% Normalisation of the different signals between 0.1 and 0.9
[du_N,a_du,b_du] = normalis(du,0.1,0.9);
[dy_N,a_dy,b_dy] = normalis(dy,0.1,0.9);
[u_N,a_u,b_u] = normalis(u,0.1,0.9);
[y_N,a_y,b_y] = normalis(y,0.1,0.9);

% saving under MAT file : 'signals.mat'
io_N1 = [dy_N;y_N;u_N;du_N]; save signals io_N1

% a and b normalisation coefficients of the inputs and outputs
save ab_norml a_du b_du a_dy b_dy a_y b_y a_u b_u
clear all
```

使用 normalis.m 文件来标准化在 alpha 和 beta 之间的信号，以使它返回线性标准化系数 a 和 b。

normalis.m file

```
function [xN,a,b] = normalis(x,alpha,beta)
a = (beta-alpha)/(max(x)-min(x));
b = beta-a*max(x);
xN = a*x+b;
```

在文件 1earn1.m 中使用误差反向传播方法来训练神经网络。权值和偏移量使用分布在 −0.5~0.5 之间的随机函数进行初始化。自适应增益为 0.8。所有的输入和输出数据文件提交给神经网络进行 300 次迭代。我们选择使用 7 个隐层。

learn1.m file

```
% network training, the output is the control variation
clear all; close all

% reading normalised io file
load signals
dy_N = io_N1(1,:); y_N = io_N1(2,:);
u_N = io_N1(3,:); du_N = io_N1(4,:);

% input cells number (6), hidden(7), output (1)
Nb_Iu = 6; Nb_Hu = 7; Nb_Ou = 1;

% random initialisation of W, Z weights, and w0, z0 biases
W = rand(Nb_Hu,Nb_Iu)-0.5; w0 = rand(Nb_Hu,1)-0.5;
Z = rand(Nb_Ou,Nb_Hu)-0.5; z0 = rand(Nb_Ou,1)-0.5 ;
```

```
N_iter = 300; % number of iterations
eta = 0.8;    % training gain
k = 0;
while k<=N_iter
   k = k+1;

   for i = 2:length(io_N1)-1
     % stimulus n° i
     x=[dy_N(i+1) dy_N(i) y_N(i) y_N(i-1) u_N(i-1) du_N(i-1)]';

     % hidden cells responses
     h = logsig(W*x+w0);

     % output cells responses
     o(i) = logsig(Z*h+z0);

     % output error
     e(i) = du_N(i)-o(i);
     % back-propagation of the error
     delta(i) = o(i).*(ones(Nb_Ou,1)-o(i)).*e(i);

     % Z Weights and z0 bias updating
     Z = Z+eta*delta(i)*h';
     z0 = z0+eta*delta(i);

     % error at the hidden cells output
     dh = h.*(ones(size(h))-h).*(Z'*delta(i));

     % W Weights and w0 biases updating
     W = W+eta*dh*x';
     w0 = w0+eta*dh;
  end
   weights_W(:,:,k) = W;
   weights_Z(:,:,k) = Z;
   bias_w0(:,:,k) = w0;
   bias_z0(:,:,k) = z0;

   % saving of the weights and biases under weights1.mat
   save weights1 weights_W weights_Z bias_w0 bias_z0

   home, clc, disp(['k = ' num2str(k)])
end
```

权值和偏移量的矩阵以多维数组的形式被保存在 *weights1. mat* 文件里。其中每一页代表每一次迭代获得的结果。

从 *read_weights.m* 文件中得到：

- 权值 W_{1k} (k=1~5)的变化连接第一输入层到所有隐层；
- 所有隐层节点的偏移量；
- 所有矩阵 Z 和偏移量元素 $z0$。

read_weights.m file

```
clear all
load weights1
%load weights2

N = length(weights_W)
W = weights_W(:,:,N), w0 = bias_w0(:,:,N)
Z = weights_Z(:,:,N), z0 = bias_z0(:,:,N)

% weights W11, W12, W13 and W14 evolution
Nb_Iu = 6; Nb_Hu = 7; Nb_Ou = 1;

for k = 1:Nb_Hu
  x = weights_W(k,1,1:100);
  plot(x(:))
  hold on
end
hold off, grid, title('weights W1k evolution')
xlabel('iterations')
% hidden layer biases evolution
figure(2)
for k = 1:Nb_Hu
  x = bias_w0(k,1,1:100);
  plot(x(:)), hold on
end
hold off, grid
title('hidden layer biases evolution ')
xlabel('iterations')

% weights Z evolution
figure(3)
for k = 1:Nb_Hu
  x = weights_Z(1,k,1:100);
  plot(x(:)), hold on
end
hold off, grid
title('weights Z evolution ')
xlabel('iterations')

% drawing of the z0 bias evolution
figure(4)
x = bias_z0(1:100);
plot(x(:)), grid
title('output cell bias evolution')
xlabel('iterations')
```

在训练结束时，获取如下权值 W、Z 和偏移量矩阵 $w0$、$z0$。

使用随机函数初始化权值和偏移量的值，就无法在每一次训练时获得相同的终值。

```
W= 1.8401  -0.9906  -0.0836   0.0543  -0.2065  -0.4587
  -2.8471   1.4276  -0.3015   0.0889   0.1165   1.0647
   1.6605  -0.8046   0.5309  -0.3332  -0.4894  -0.5976
```

```
   -0.1242    0.2374    0.4310   -0.3364    0.2788   -0.1862
    0.9671    0.0443    0.0021   -0.2165   -0.0323   -0.0311
   -1.4171    1.0036    0.0788   -0.5783    0.3564    0.7720
    1.5023   -0.3170   -0.1589    0.3691    0.0271   -0.8963
Wo=0.0778
    0.3113
    0.1729
    0.1265
    0.2154
   -0.1377
   -0.0746
z=
    1.9110   -3.6989   1.8310   -0.2640   0.6325   -2.0627
Zo=
   -0.2265
```

每一次迭代训练时，权值和偏移量曲线如图 4-26 至图 4-28 所示。

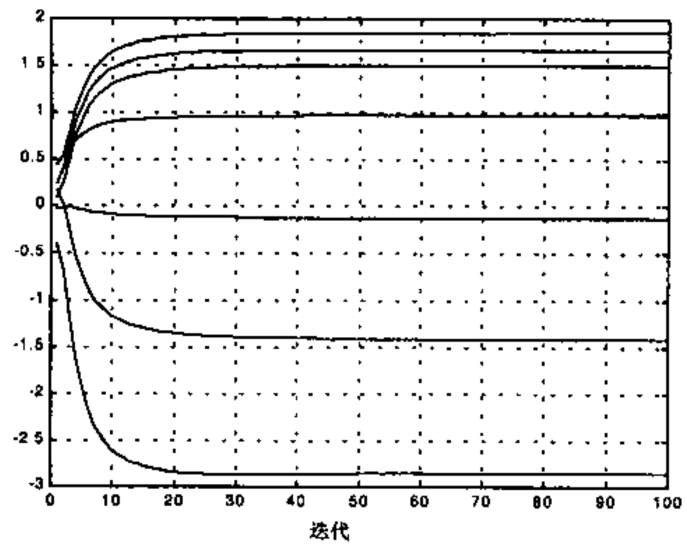


图 4-26　权值 W 变化曲线

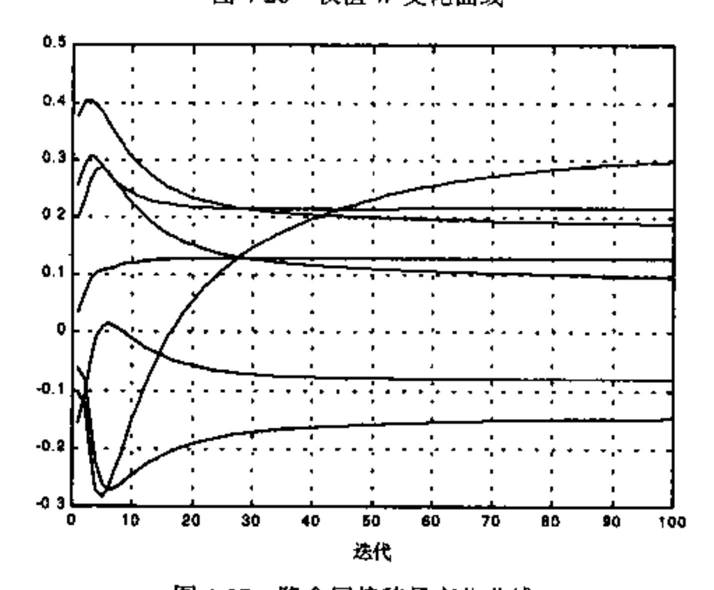


图 4-27　隐含层偏移量变化曲线

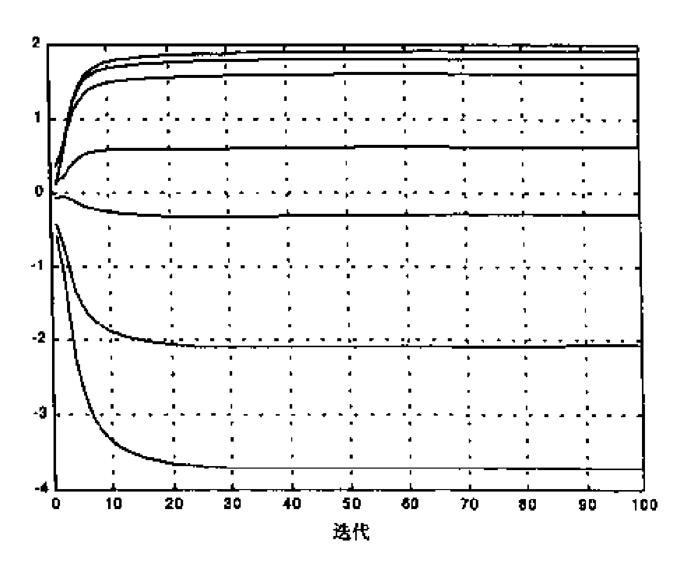


图 4-28　权值 Z 变化曲线

在大约 50 次迭代后，可以得到一个很好的权值和偏移量的收敛曲线。尽管自适应增益的值有足够的提高，但权值和偏移量在 100 次迭代之后就基本不再变化了（见图 4-29）。

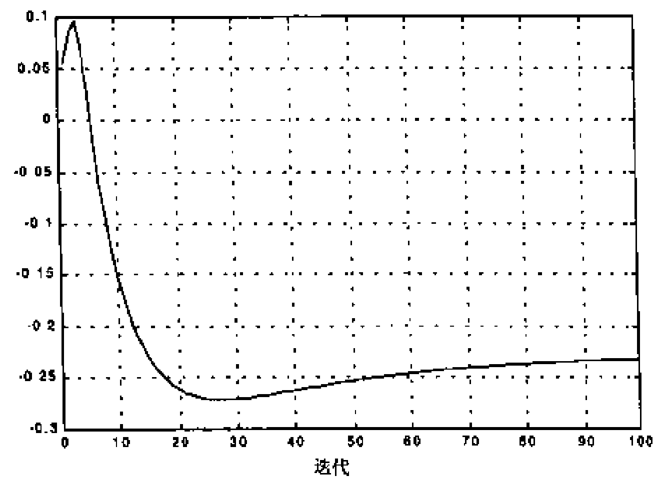


图 4-29　输出层偏移量变化曲线

为了验证训练的质量，我们把同样的标准化信号应用到权值和偏移量固定为最终值的神经网络。与文件中网络输出的标准化控制变量比较，可以显示逆模型识别的质量。

verif_train.m file

```
clear all; close all

% reading the normalised IO data file
load signals
dy_N = io_N1(1,:);
y_N = io_N1(2,:);
u_N = io_N1(3,:);
du_N = io_N1(4,:);

% reading the weights and biases data file
load weights1
```

```
%N = length(weights_W)-1
N=5;
W = weights_W(:,:,N)
w0 = bias_w0(:,:,N)
Z = weights_Z(:,:,N)
z0 = bias_z0(:,:,N)
clear weights_W weights_Z bias_w0 bias_z0

  for i = 2:length(io_N1)-1
    % stimulus n° i
    x=[dy_N(i+1) dy_N(i) y_N(i) y_N(i-1) u_N(i-1) du_N(i-1)]';

    % output layer response
    y(i) = logsig(Z*(logsig(W*x+w0))+z0);

  end % boucle for

y = y(1:100);
figure(1), plot(y), hold
h = plot(du_N(1:100));
set(h,'LineWidth',1.5)
title(['real and desired network responses after ' num2str(N) ' iterations'])
axis([0 100 0 1]), grid, hold off

figure(2), du_N = du_N(1:100);
error = y-du_N;
plot(error), grid
axis([0 100 -0.2 0.2])
title(['training error ' 'after ' num2str(N) ' iterations'])

disp(['maximal error : ' num2str(max(error))])
disp(['error variance : ' num2str(std(error)^2)])
```

在大约 300 次迭代后，可以获得实际网络响应和期望信号的极其相似性（见图 4-30）。

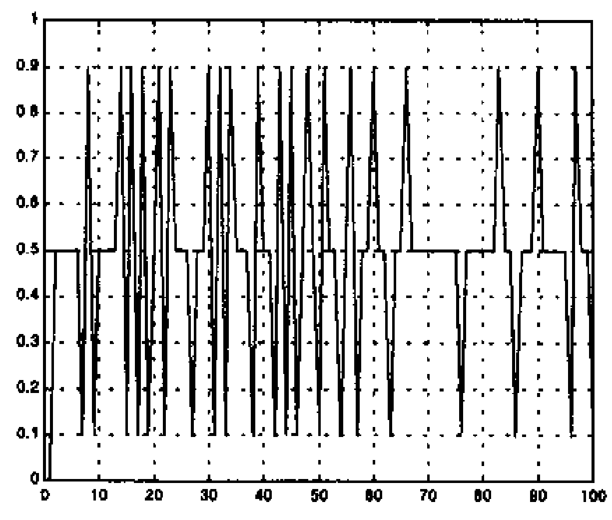


图 4-30　300 次迭代后实际网络输出和期望信号

图 4-31 所示的曲线表示训练误差的变化。

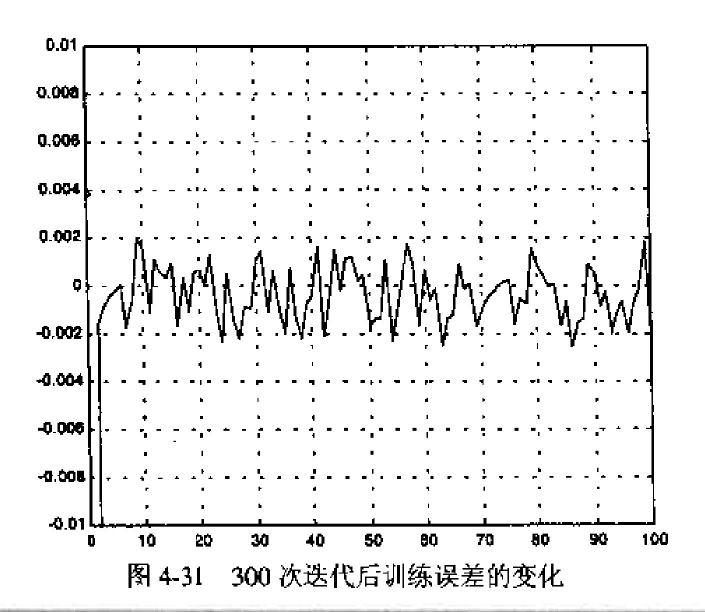


图 4-31　300 次迭代后训练误差的变化

```
maximal error    :  0.0019054
error variance   :  0.0024975
```

在大约仅仅 5 次迭代(文件 verif_train.m 中 N=5)后，可以得到：

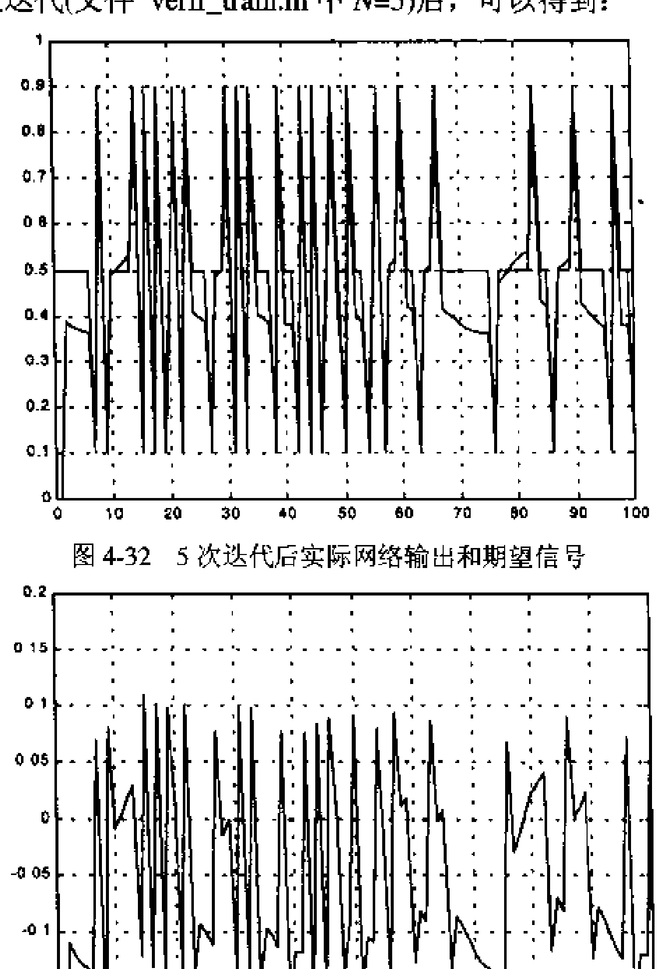


图 4-32　5 次迭代后实际网络输出和期望信号

图 4-33　5 次迭代后训练误差的变化

```
maximal error    :  0.10936
error variance   :  0.0099786
```

为了实现一个逆模型神经网络系统，第一层节点的输入为期望目标 $r(t+1)$和过程实际输出 $y(t)$值的差的标准化值（见图 4-34）。

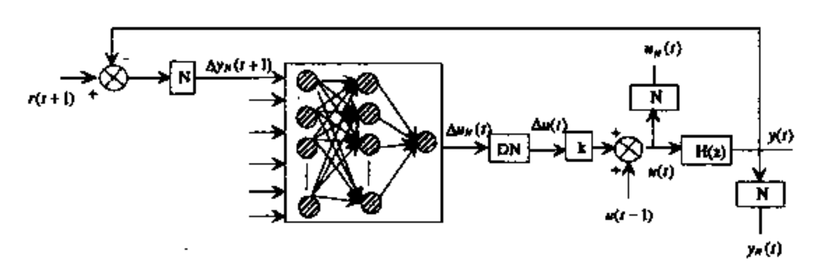


图 4-34　系统结构框图

文件 *cde_nrl.m* 实现了上面所画的控制律，这个神经网络的输出信号去除标准化并乘以增益 k 后叠加到前一个采样时刻的控制信号。增益 k 对闭环稳定性有影响。

由于积分控制的存在，稳态误差为零。所以，可以使用一个带有积分环节的预测控制。

ctr_nrl.m

```
% inverse neural model control
clear all; close all, clc
% reading the weights
load weights1
N = length(weights_W);
W = weights_W(:,:,N);
w0 = bias_w0(:,:,N);
Z = weights_Z(:,:,N);
z0 = bias_z0(:,:,N);
clear weights_W weights_Z bias_w0 bias_z0
% reading of normalisation coefficients
load ab_norm1
% target signal generation
stair1 = 5*ones(1,100); stair2 = 6*ones(1,100);
r1 = [stair1 stair2];
t = 0:pi/100:2*pi;
r2 = 6+sin(0.8*t);
r = [r1 r2]; y = r; % model output initialisation
u = 5*ones(size(r)); y_N = a_y*y+b_y;
du_N = zeros(size(r)); u_N = a_u*u+b_u;
  for i = 3:length(r)-1
      % process output
      y(i) = 0.7*y(i-1)+0.3*u(i-1)+0.05*u(i-2);
      % error normalisation
      err = a_dy*(r(i+1)-y(i))+b_dy;
      % output variation disnormalisation
      dy_N = a_dy*(y(i)-y(i-1))+b_dy;
      % output normalisation
      y_N(i) = a_y*y(i)+b_y;
      % stimulus n° i
      x = [err dy_N y_N(i) y_N(i-1) u_N(i-1) du_N(i-1)]';
```

```
    % hidden layer's cells activation
    h_in = W*x+w0;
    h_in = h_in.*(-5<=h_in<=5)-5.*(h_in<-5)+5.*(h_in>5);
    % outputs of hidden layer's cells
    h_out = logsig(h_in);
    % hidden layer's cells activation
    o_in = Z*h_out+z0;
    o_in = o_in.*(-5<=o_in<=5)-5.*(o_in<-5)+5.*(o_in>5);
    % hidden layer's cells outputs
    du_N(i) = logsig(o_in);
    % hidden layer's cells response
    u(i) = ((du_N(i)-b_du)/a_du)*0.1+u(i-1);
    u_N(i) = a_u*u(i)+b_u;
    % perturbation at k=300 discrete time
    y(i) = y(i)-(i==300)/2;
end  % for loop
% drawing different signals
figure(1), plot(y), hold, plot(r,'-.')
title('target and output signals')
xlabel('discrete time'), axis([0 400 4.5 7.2])
text(110,5.7,'\leftarrow output signal',    'HorizontalAlignment','left')
figure(2), stairs(u), title('control signal')
xlabel('discrete time'), axis([0 400 4 6.2]), grid
```

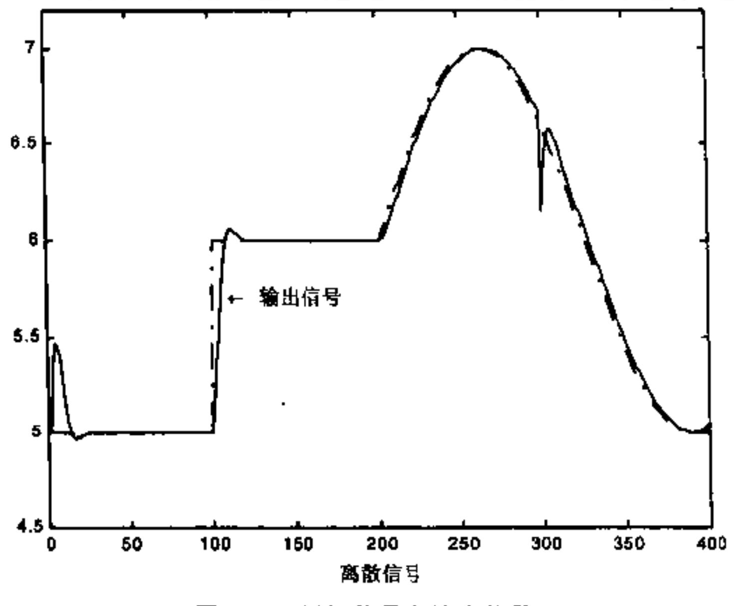


图 4-35　目标信号和输出信号

在离散时刻 t=300 时，可以在过程输出中加入一个干扰，它产生了一个小的超调（见图 4-36）。

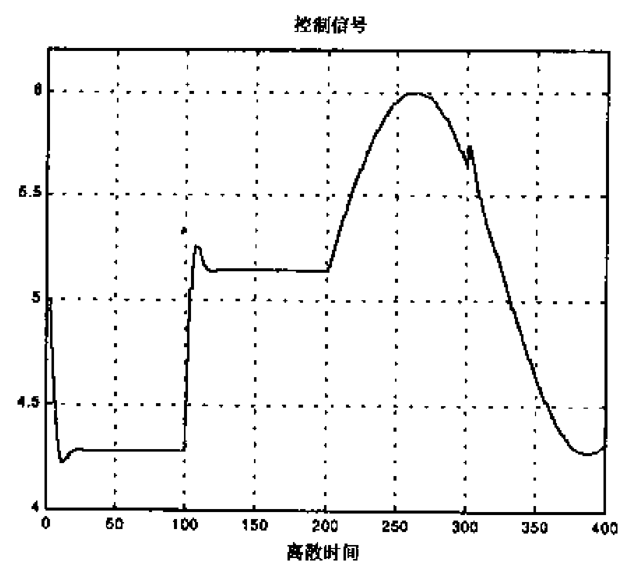


图 4-36 控制信号

◆ SIMULINK 模型（见图 4-37）

在仿真以前，必须顺序执行文件 read_weights.m，并读入数据文件 ab-norml.mat，以便恢复权值和偏移量矩阵，以及输入、输出信号的标准化系数。

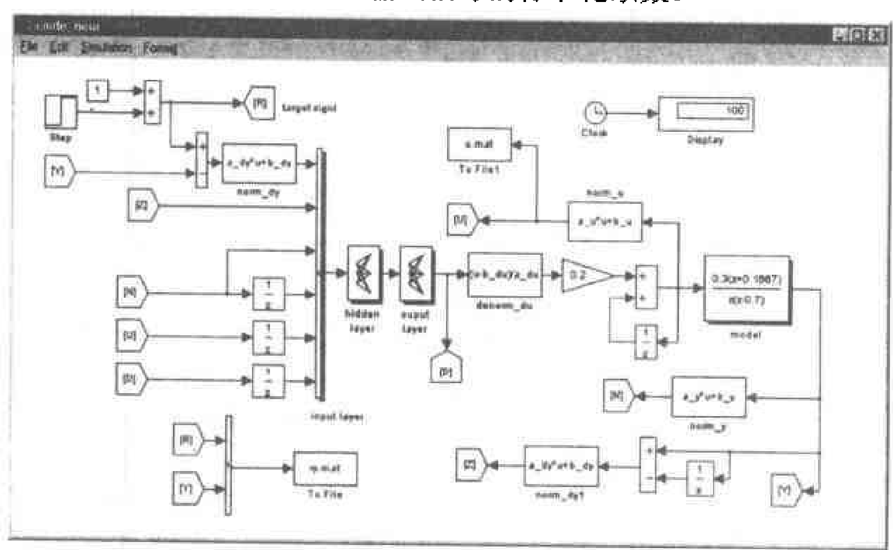


图 4-37 SIMULINK 模型

文件 result_sim.m 可以恢复控制信号、目标信号和输出信号。

result_sim.m file

```
% displaying the control results
% with inverse neural model in SIMULINK

% recupération of the time and the control signal
load u.mat
t = u(1,:); u = u(2,:);

% drawing control signal
stairs(t,u), xlabel('discrete time')
title('control signal'), grid
```

```
% recupération of the target and output signals
load ry.mat
r = ry(2,:); y = ry(3,:);

% drawing target and output signals
figure(2)
plot(t,r,:), hold on, plot(t,y)
text(53,1.5,'\leftarrow output signal',...
    'HorizontalAlignment','left')
xlabel('discrete time')
title('target and output signals'), hold off
```

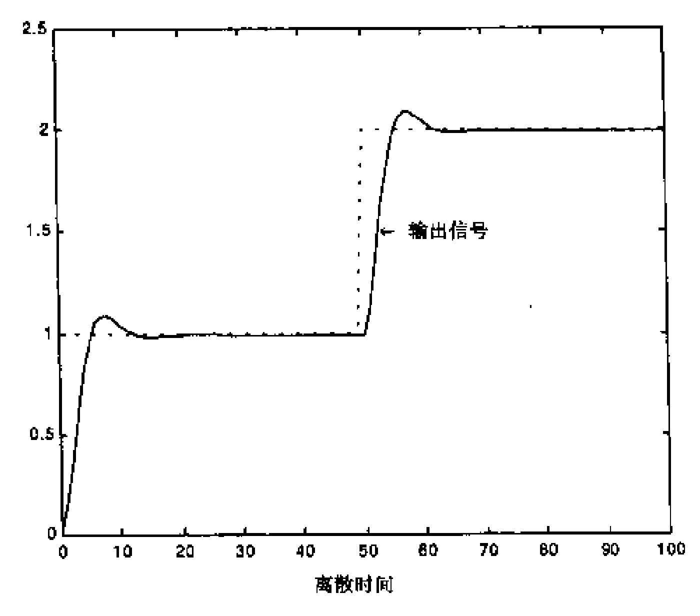


图 4-38　目标信号和输出信号

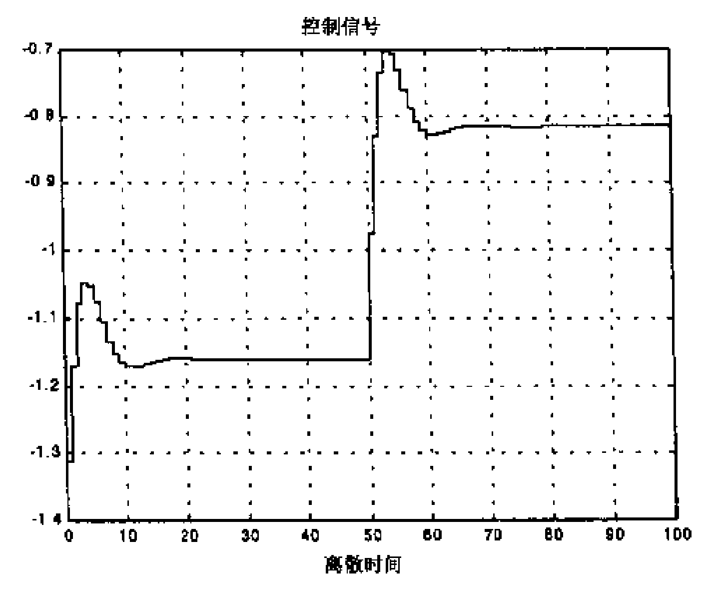


图 4-39　控制信号

4.4.2 第二层网络结构

另一种可得到过程模型的逆模式神经网络结构如图 4-40 所示。

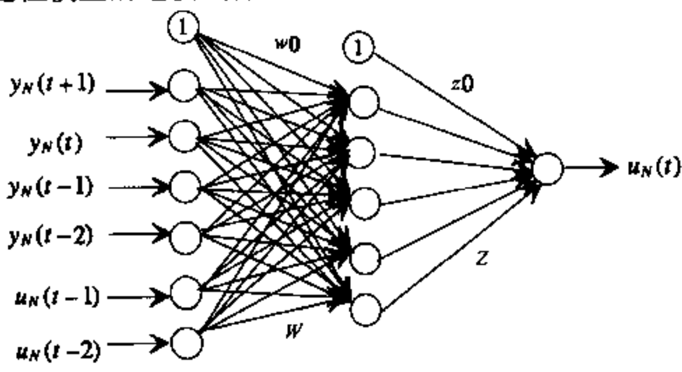


图 4-40 第二层网络结构

输入节点直接接收标准化信号，在网络输出端设法得到标准化控制信号。不过，仍需保持控制律的预测特性，标准化的预测控制信号应用于第一个输入层的节点。执行 model.m 文件来产生控制信号的新值和输出模型。

文件 io_norm2.m 能够标准化不同的信号并把它们保存在数据文件 ic_N2.mat 中。标准化系数保存在数据文件 ab_norm2.mat 中。

io_norm2.m file

```
% different signals normalisation between 0.1 and 0.9
[u_N,a_u,b_u] = normalis(u,0.1,0.9);
[y_N,a_y,b_y] = normalis(y,0.1,0.9);

% saving signals under 'signaux.mat' data file
io_N2 = [u_N;y_N];
save signals io_N2

% a and b normalisation coefficients of I/O
save ab_norm2 a_u b_u a_y b_y
clear all
```

在文件 learn2.m 中进行神经网络的训练。

learn2.m file

```
% network learning
clear all, close all

% reading the normalised I/O data file
load signals
u_N = io_N2(1,:);
y_N = io_N2(2,:);

% number of input cells (6), hidden(7) and output (1)
Nb_Iu = 6; Nb_Hu = 7; Nb_Ou = 1;

% random initialisation of the weights W, Z and biases w0,z0
W = rand(Nb_Hu,Nb_Iu)-0.5; w0 = rand(Nb_Hu,1)-0.5;
```

```
Z = rand(Nb_Ou,Nb_Hu)-0.5; z0 = rand(Nb_Ou,1)-0.5;

N_iter = 300; % number of iterations
eta = 0.8;    % learning gain
k = 0;
while k<=N_iter
  k = k+1;
  for i = 3:length(io_N2)-1
    % stimulus n° i
    x=[y_N(i+1) y_N(i) y_N(i-1) y_N(i-2) u_N(i-1) u_N(i-2)]';

    % hidden layer's cells response
    h = logsig(W*x+w0);

    % output layer's cells response
    o(i) = logsig(Z*h+z0);
    % error at network output
    e(i) = u_N(i)-o(i);

    % error to be backpropagated
    delta(i) = o(i).*(ones(Nb_Ou,1)-o(i)).*e(i);

    % updating of Z weights and bias z0 matrices
    Z = Z+eta*delta(i)*h'; z0 = z0+eta*delta(i);

    % error at hidden layer output
    dh = h.*(ones(size(h))-h).*(Z'*delta(i));

    % updating of W weights and bias w0 matrices
    W = W+eta*dh*x';
    w0 = w0+eta*dh;

  end % for
  weights_W(:,:,k) = W; weights_Z(:,:,k) = Z;
  bias_w0(:,:,k) = w0; bias_z0(:,:,k) = z0;

  % saving the weights and biases under weights.mat data file
  save weights2 weights_W weights_Z bias_w0 bias_z0

  clc, disp(['k = ' num2str(k)])
end % while loop
```

执行文件 read_weights.m，用“loadweights2”代替第一个命令行“loadweights1”，显示出不同的权值和偏移量矩阵。

```
W=
   2.8411    0.0614   -0.2703   -0.3327   -0.7980   -0.4991
   6.1985   -0.6925   -1.2043   -1.0801   -1.1725   -0.4669
  -0.3686    0.2813    0.6997    0.4166    0.5809    0.0158
  -2.1526    0.5725    0.4382    0.2168    0.0058    0.1429
   1.7550   -0.2303   -0.3786   -0.4863   -0.4626   -0.1773
```

```
  -0.3108    0.3688    0.1481    -0.0237    0.5202    0.1999
  -5.4864    1.2625    0.9630     0.6244    0.9624    0.2813
w0=
   -0.2506
   -0.6902
    0.2817
    0.2824
    0.2624
    0.5206
    0.6028
z=
    2.5569    6.0432    -0.8098    -2.3561    1.5498    -0.6587    -5.7637
z0=
   -0.5345
```

权值和偏移量的变化曲线如图 4-41 至图 4-44 所示，和前面的例子一样，在约 40 次迭代后收敛。

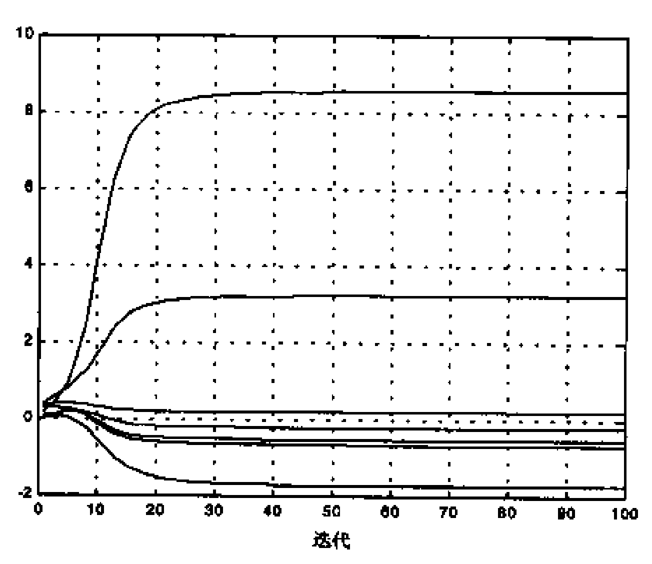


图 4-41　权值 W 变化曲线

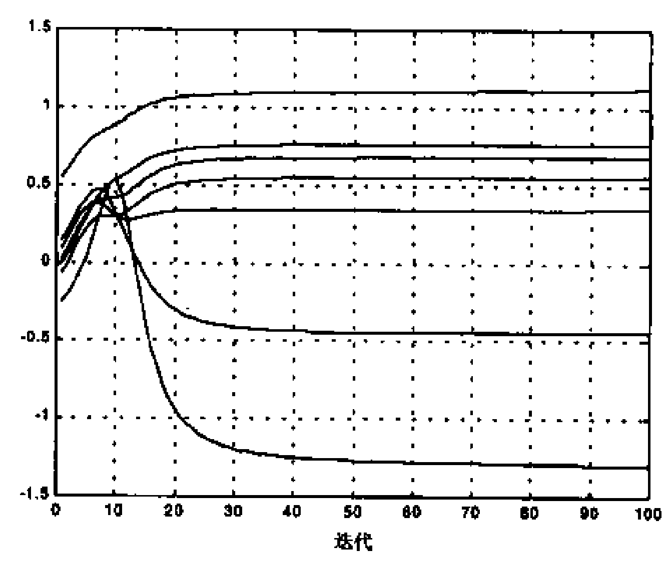


图 4-42　隐含层偏移量变化曲线

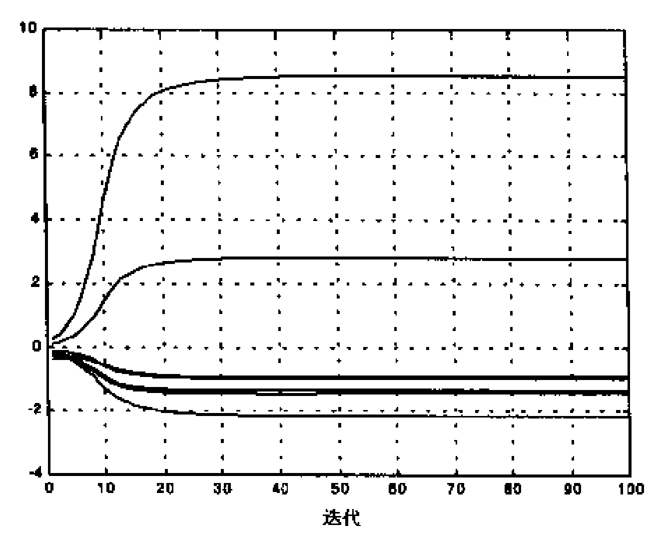


图 4-43　权值 Z 变化曲线

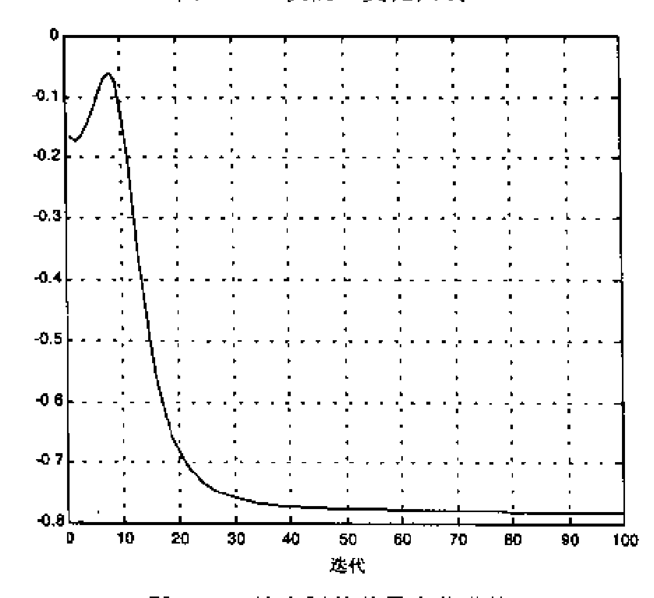


图 4-44　输出层偏移量变化曲线

下面检验第一个网络结构的学习结果。

verify_learn2.m file

```
% network learning
clear all; close all

% reading normalised I/O data file
load signals
u_N = io_N2(1,:);
y_N = io_N2(2,:);

% reading weights and biases data file
load weights2

N = 100;
W = weights_W(:,:,N), w0 = bias_w0(:,:,N)
Z = weights_Z(:,:,N), z0 = bias_z0(:,:,N)
```

```
clear weights_W weights_Z bias_w0 bias_z0

  for i = 3:length(io_N2)-1
    % stimulus n° i
    x=[y_N(i+1) y_N(i) y_N(i-1) y_N(i-2) u_N(i-1) u_N(i-2)]';

    % output layer's cells response
    y(i) = logsig(Z*(logsig(W*x+w0))+z0);

  end % for loop

y = y(1:100);
figure(1)
stairs(y)
hold
h = stairs(u_N(1:100));
set(h,'LineWidth',1.5)
title(['real and desired network responses after '...
      num2str(N) ' iterations'])
axis([0 100 0 1])
grid
hold off

figure(2)
u_N = u_N(1:100);
error = y-u_N;
plot(error)
grid
axis([0 100 -0.02 0.02])
title(['learning error ' ' after ' num2str(N)...
      ' iterations'])

disp(['maximal error  : ' num2str(max(error))])
disp(['error variance : ' num2str(std(error)^2)])
```

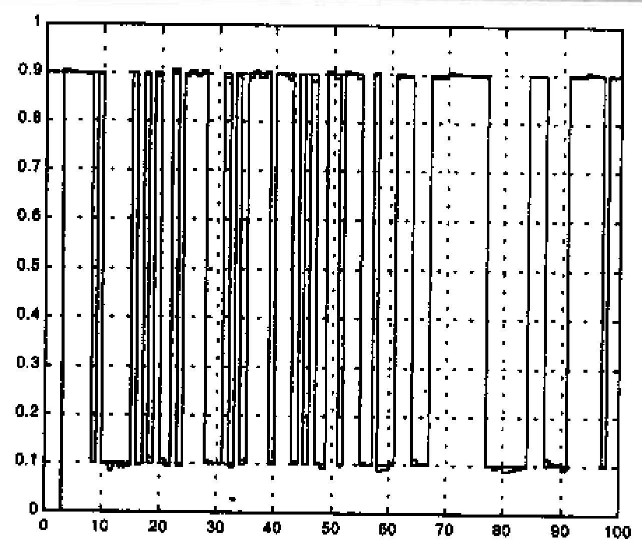


图 4-45　100 次迭代后实际网络输出和期望信号

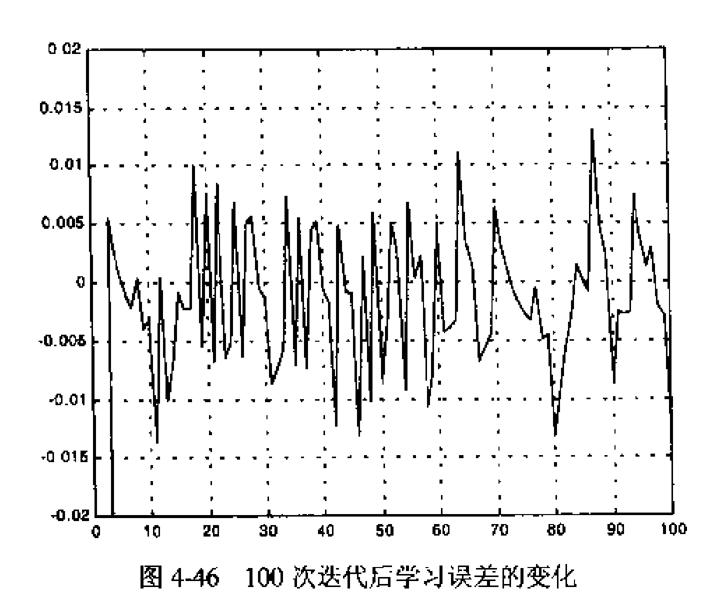


图 4-46　100 次迭代后学习误差的变化

```
maximal error    : 0.013077
error variance   : 0.016021
```

误差方差接近于它的最大值，约为 10^{-2}。误差只发生在控制信号为稳态值时，由于神经网络结构的控制律没有积分环节，在稳态域必然要产生跟踪误差。

在下面的控制律中，第一个输入层的节点接收控制信号的预测值，如前面的例子所示，这样可以消除输出信号相对于目标信号的一步延迟。

在去除标准化后，神经网络的响应将构成能够直接应用于过程的控制信号（见图 4-47）。

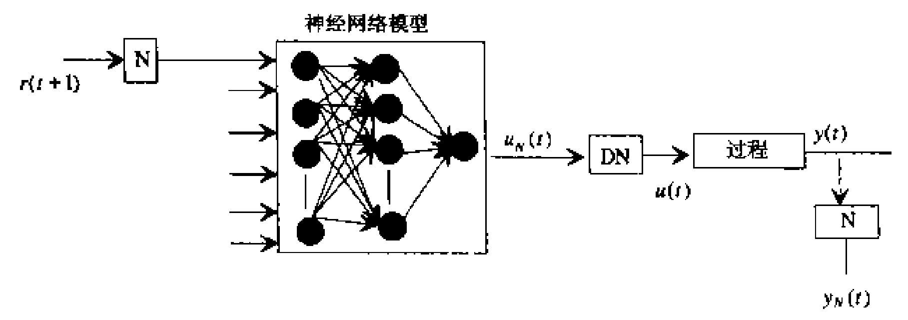


图 4-47　控制系统结构框图

文件 *ctr_nr2.m* 将这个控制律应用于这种类型的过程模型。

ctr_nr2.m file

```
% inverse neural model control
clear all, close all, clc

% reading the weights
load weights2
N = length(weights_W);
W = weights_W(:,:,N);
w0 = bias_w0(:,:,N);
Z = weights_Z(:,:,N);
z0 = bias_z0(:,:,N);
clear weights_W weights_Z bias_w0 bias_z0

% reading of normalisation coefficients
```

```
load ab_norm2

% target signal generation
stair1 = 5*ones(1,100);
stair2 = 6*ones(1,100);
r1 = [stair1 stair2];

t = 0:pi/100:2*pi;
r2 = 6+sin(t);
r = [r1 r2];
y = r; % initialisation de la sortie du modèle
u = 5*ones(size(r));
y_N = a_y*y+b_y;
u_N = a_u*u+b_u;

  for i = 3:length(r)-1

    % process output
    y(i) = 0.7*y(i-1)+0.3*u(i-1)+0.05*u(i-2);

    % error normalisation
    r_N = a_y*r(i+1)+b_y;

    % normalisation de la sortie actuelle
    y_N(i) = a_y*y(i)+b_y;

    % stimulus n° 1
    x=[r_N y_N(i) y_N(i-1) y_N(i-2) u_N(i-1) u_N(i-2)]';

    % hidden layer's cells activation
    h_in = W*x+w0;
    h_in = h_in.*(-5<=h_in<=5)-5.*(h_in<-5)+5.*(h_in>5);

    % outputs of hidden layer's cells
    h_out = logsig(h_in);

    % output layer's cells activation
    o_in = Z*h_out+z0;
    o_in = o_in.*(-5<=o_in<=5)-5.*(o_in<-5)+5.*(o_in>5);

    % output layern's cells response
    u_N(i) = logsig(o_in);

    % control signal de-normalisation
    u(i) = (u_N(i)-b_u)/a_u;

    % creation of a disturbance at k = 300
    y(i) = y(i)-(i==300)/2;
  end % for loop

% plotting the different signals
```

误差。

```
figure(1), plot(y), hold, plot(r,'-.')
text(296,5.7,'output disturbance \rightarrow',...
    'HorizontalAlignment','right')
title('target and output signals'), xlabel('discrete time')
axis([0 400 4.5 7.2])

figure(2), stairs(u)
title('control signal')
xlabel('discrete time'), axis([0 400 4 6.2])
grid
```

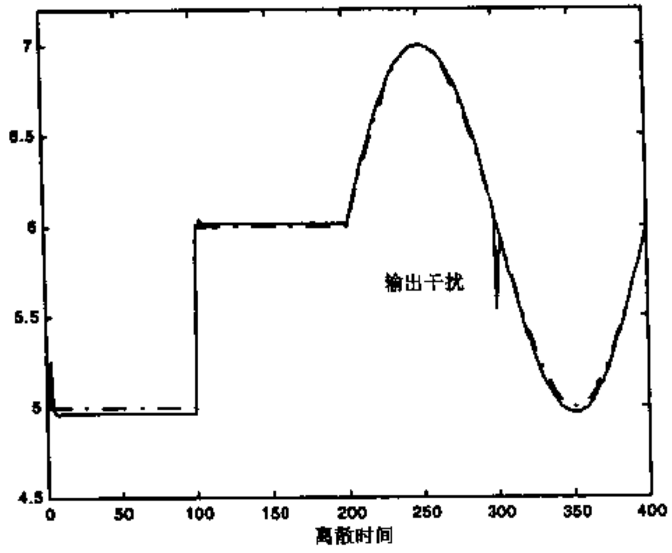


图 4-48　目标信号和过程输出

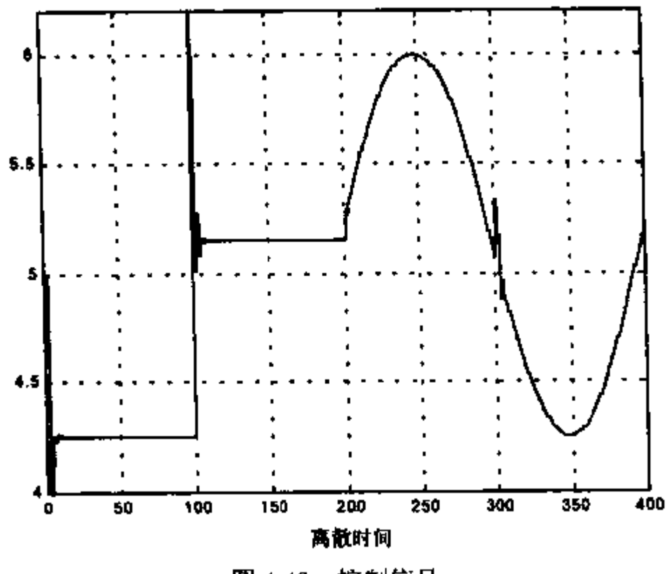


图 4-49　控制信号

由于没有积分环节，因此增加了稳态模型误差，使目标信号和过程输出之间存在稳态误差。

为减小或抑制稳态误差，可以在神经网络控制的基础上增加一个积分控制，或者在每个采样时刻更新权值和偏移量（自适应控制）。

4.4.2.1 增加积分环节

在神经网络控制的基础上增加的控制由跟踪误差的积分产生（见图 4-50）。

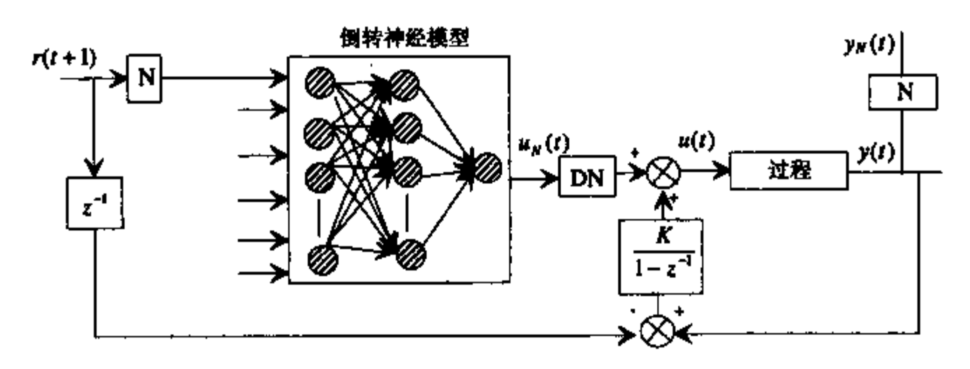


图 4-50　系统结构框图

在神经网络提供的控制上增加了一个增益为 0.05 的积分控制。同时刻的目标和控制信号保存在数据文件 cs.dat 中，控制信号保存在数据文件 ctr.mat 中（见图 4-51）。

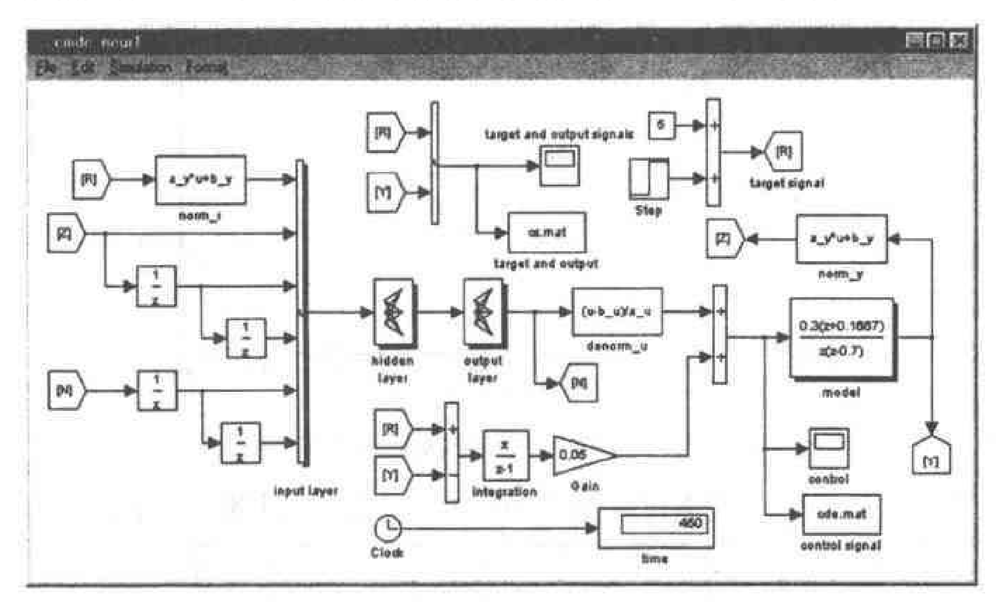


图 4-51　SIMULINK 仿真结构图

文件 read_file.m 可以读不同的文件并显示控制和输出信号。

read_file.m file

```
% reading the I/O files
load cs
t = cs(1,:);
r = cs(2,:);
y = cs(3,:);

% displaying the target and the output
plot(t,r,':'), hold on, plot(t,y)
axis([50 450 4.5 6.5])
title('inverse neural model control plus integration')
xlabel('discrete time'), hold off
```

```
% file of control signal
load cde
u = cde(2,:);

% displaying the control signal
figure(2)
stairs(t,u)
title('control signal')
xlabel('discrete time')
axis([50 450 4 7])
xlabel('discrete time')
```

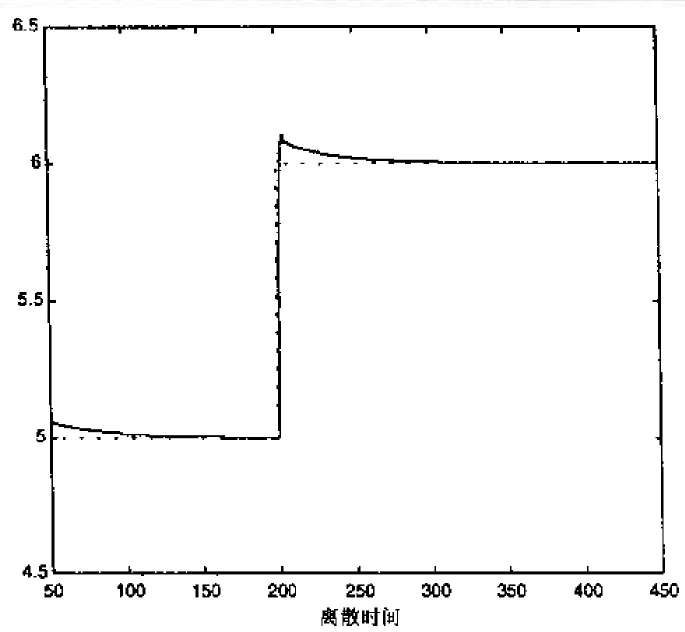


图 4-52 带积分环节的逆模式神经网络控制

由于增加了积分环节，稳态误差被抑制掉。此外，增益值也影响闭环系统的时间响应。

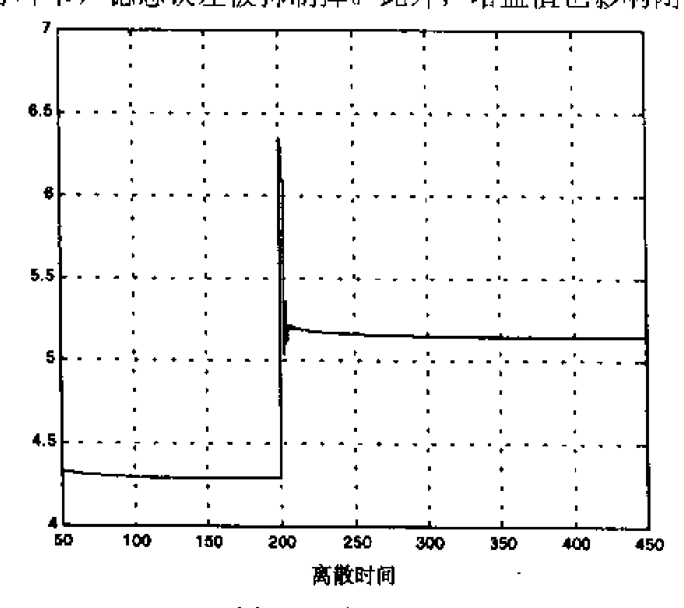


图 4-53 控制信号

4.4.2.2 自适应控制

在每个采样时刻把跟踪误差反向传播到神经网络，修改权值和偏移量来消除稳态误差。这意味着必须不断学习，以便消除稳态误差。由于这时误差是静态的，所以偏移量会有很大变化（见图 4-54）。

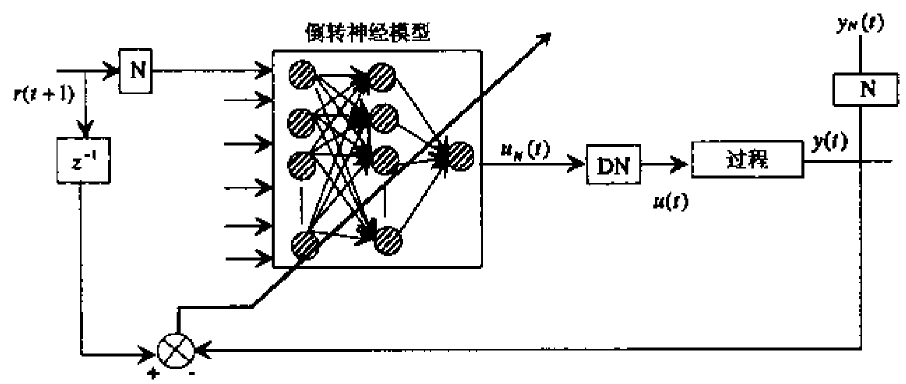


图 4-54 自适应控制结构框图

在文件 ctr_nr3.m 中实现了这种控制，包括预测和自适应控制。

ctr_nr3.m file

```
% inverse neural network control
% with weights adaptation in order to cancel the steady state
% error
clear all, close all
clc
% reading the weights
load weights2
N = length(weights_W);
W = weights_W(:,:,N);
w0 = bias_w0(:,:,N);
Z = weights_Z(:,:,N);
z0 = bias_z0(:,:,N);
clear weights_W weights_Z bias_w0 bias_z0

% reading the normalisation coefficients
load ab_norm2

% weights adaptation gain
%eta = 0.2;
eta = 0.8;


% target signal generation
stair1 = 5*ones(1,100);
stair2 = 6*ones(1,100);
r1 = [stair1 stair2];

t = 0:pi/100:2*pi;
r2 = 6+sin(t);
r = [r1 r2];
```

```
y = r; % model output initialisation
u = 5*ones(size(r));
y_N = a_y*y+b_y;
u_N = a_u*u+b_u;

% weights and biases initial values
weights_W = cat(3,W,W,W);
bias_w0 = cat(3,w0,w0,w0);
weights_Z = cat(3,Z,Z,Z);
bias_z0 = cat(3,z0,z0,z0);

  for i = 3:length(r)-1
    % reading of the process output
    y(i) = 0.7*y(i-1)+0.3*u(i-1)+0.05*u(i-2);

    % future target signal value normalisation
    r_N = a_y*r(i+1)+b_y;

    % actual output normalisation
    y_N(i) = a_y*y(i)+b_y;
    % stimulus n° i
    x = [r_N y_N(i) y_N(i-1) y_N(i-2) u_N(i-1) u_N(i-2)]';

    % hidden layer's cells activation
    h_in = W*x+w0;
    h_in = h_in.*(-5<=h_in<=5)-5.*(h_in<-5)+5.*(h_in>5);

    % hidden layer's cells output
    h = logsig(h_in);

    % output layer's cells activation
    o_in = Z*h+z0;
    o_in = o_in.*(-5<=o_in<=5)-5.*(o_in<-5)+5.*(o_in>5);

    % output layer's cells response
    o = logsig(o_in);
    u_N(i) = o;

    % control signal disnormalisation
    u(i) = (u_N(i)-b_u)/a_u;

    % creation of a disturbance at k = 300
    y(i) = y(i)-(i==300)/2;

    % updating weights and biases
    e = r(i)-y(i);

    % error to backpropagate
    delta = o.*(ones(1,1)-o).*e;

    % updating weights Z and bias z0 matrices
    Z = Z+eta*delta*h';
    z0 = z0+eta*delta;
```

```
    % error in hidden layer output
    dh = h.*(ones(size(h))-h).*(Z'*delta);

    % updating weights W and bias w0 matrices
    W = W+eta*dh*x';
    w0 = w0+eta*dh;

    % saving the weights and biases
    weights_W = cat(3,weights_W,W);
    bias_w0 = cat(3,bias_w0,w0);
    weights_Z = cat(3,weights_Z,Z);
    bias_z0 = cat(3,bias_z0,z0);
end % for loop

% plotting the different signals
figure(1)
plot(y), hold, plot(r,'-.')
title(['target and output, Gain = ' num2str(eta)])
xlabel('discrete time')
axis([0 400 4.5 7.2]), grid

figure(2)
stairs(u)
title(['control signal, Gain = ' num2str(eta)])
xlabel('discrete time')
axis([0 400 4 6.2]), grid

% plotting weights W11, W12, W13 and W14 evolution
figure(3)
for k = 1:5
  x = weights_W(k,1,:);
  plot(x(:)), hold on
end
hold off, grid
title('weights W1k')
xlabel('discrete time')

% plotting hidden layer biases evolution
figure(4)
for k = 1:5
  x = bias_w0(k,1,:);
  plot(x(:)), hold on
end
hold off, grid
title('hidden layer biases')
xlabel('discrete time')

% plotting weights Z evolution
figure(5)
for k = 1:5
  x = weights_Z(1,k,:);
  plot(x(:)), hold on
```

```
end
hold off, grid
title('weights 2'), xlabel('discrete time')

% plotting bias z0 evolution
figure(6)
plot(bias_z0(:)), grid
title('output cell bias')
xlabel('discrete time')
```

用在学习阶段得到的值来初始化不同的权值和偏移量。可以看到，静态误差被完全消除了，但与有积分环节的控制相比，抗干扰时有一个超调。

时间响应和超调量取决于权值和偏移量的自适应增益。自适应增益等于 0.2，抗干扰时的超调很大（见图 4-55、图 4-56）。

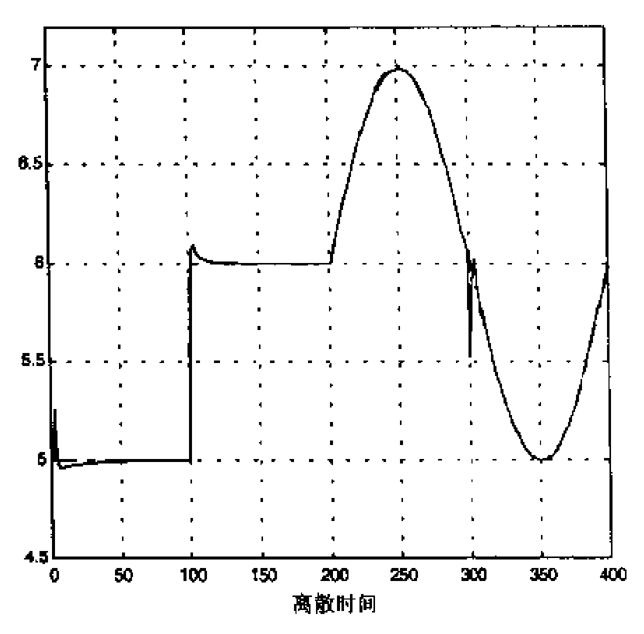


图 4-55 自适应增益等于 0.2 时的目标信号和输出信号

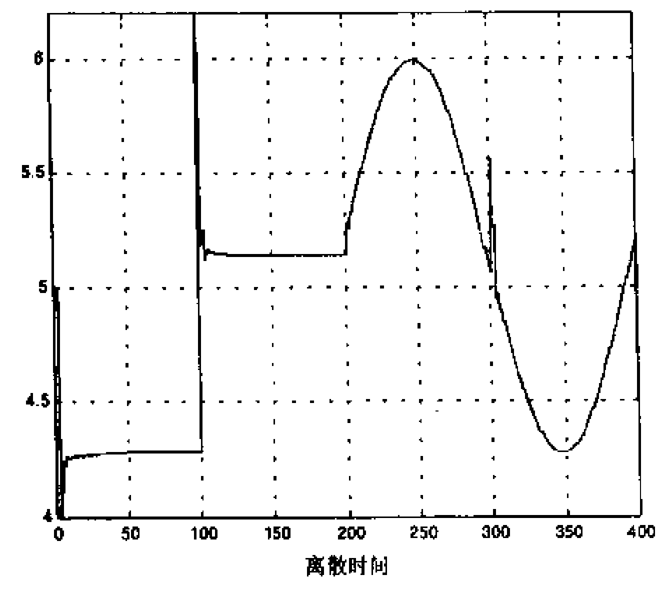


图 4-56 自适应增益等于 0.2 时的控制信号

图 4-57 到图 4-60 显示了在用学习阶段得到的值初始化后权值和偏移量的变化。

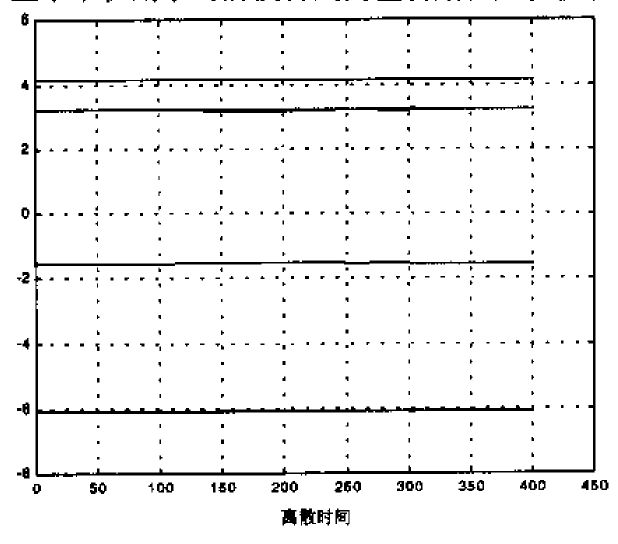


图 4-57　权值 W 的变化曲线

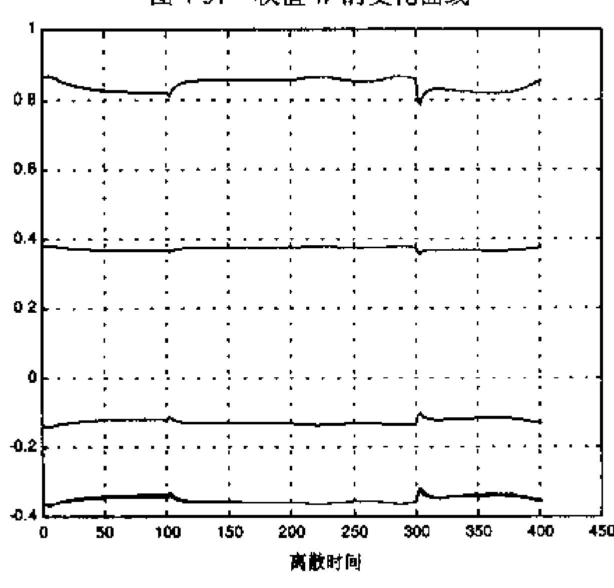


图 4-58　隐含层偏移量的变化曲线

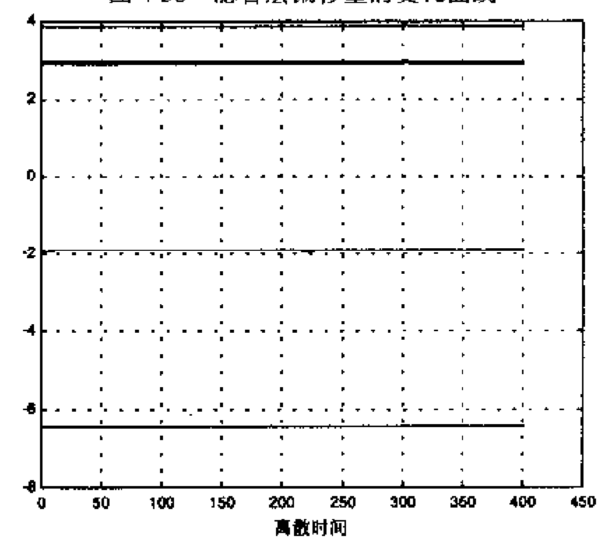


图 4-59　权值 Z 的变化曲线

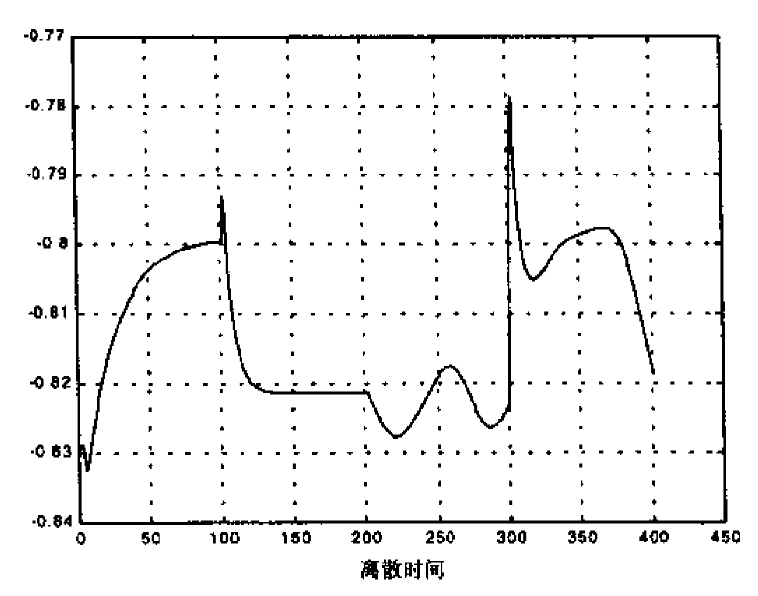


图 4-60　输出层偏移量的变化曲线

相对于偏移量，W 和 Z 的变化并不大，因为消除的误差是静态的。增益等于 0.8 时，响应时间很短，但在抗干扰时有超调（见图 4-61）。

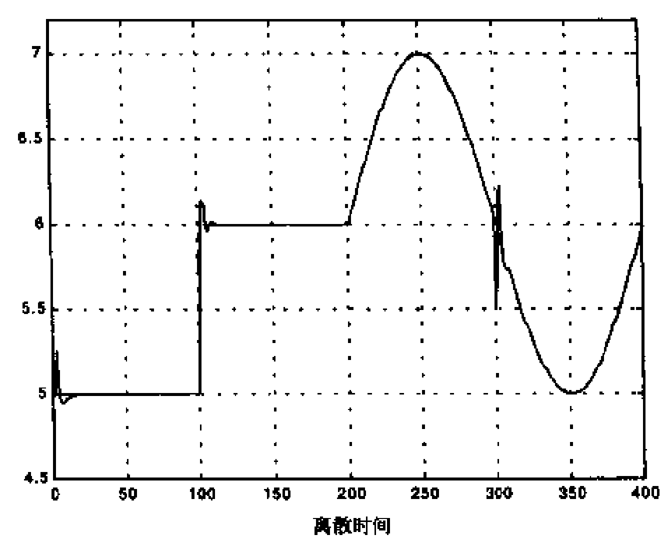


图 4-61　自适应增益等于 0.8 时的目标信号和输出信号

对于一个方波目标信号，我们指定一个参考模型来实现跟踪动态特性。如果想让过程输出跟踪目标信号，通过一个具有标准化振荡频率 $\omega_0 T$ 和阻尼比 ξ，以及单位增益的二阶系统，参考信号由下面的目标信号决定：

$$r(t)=\frac{1+\alpha_1+\alpha_2}{1+\alpha_1 z^{-1}+\alpha_2 z^{-2}}c(t)$$

其中：

$$\alpha_1=-2\mathrm{e}^{-\xi\omega_0 T}\cos(\sqrt{1-\xi^2}\,\omega_0 T)$$

$$\alpha_2=\mathrm{e}^{-2\xi\omega_0 T}$$

在文件 cde_modref.m 中，指定了一个阻尼比 $\xi=\dfrac{\sqrt{2}}{2}$ 的二阶过滤器。

cde_modref.m file

```
% neural adaptive control with reference model

% generation of target signal
close all
stair1 = 5*ones(1,100);
stair2 = 6*ones(1,100);
c = [stair1 stair2 stair1 stair2];

% reference model
dzeta = sqrt(2)/2;
w0T = 0.08*pi;

alfa1 = -2*exp(-dzeta*w0T)*cos(w0T*sqrt(1-dzeta^2));
alfa2 = exp(-2*dzeta*w0T);

r(1:2) = c(1:2);

for i = 3:length(c)
    r(i) = (1+alfa1+alfa2)*c(i)-alfa1*r(i-1)-alfa2*r(i-2);
end

% displaying the target signal
plot(r)
hold on
plot(c,'-.')

% reading weights and biases data file
load weights2
N = length(weights_W); % number of iterations

% recovering of the last page of arrays
W = weights_W(:,:,N);
w0 = bias_w0(:,:,N);
Z = weights_Z(:,:,N);
z0 = bias_z0(:,:,N);
clear weights_W weights_Z bias_w0 bias_z0

% reading the normalisation coefficients
load ab_norm2

% adaptation gain of the weights
eta = 0.5;

y = r; % output model initialisation
u = 5*ones(size(r));
y_N = a_y*y+b_y;
u_N = a_u*u+b_u;

% weights and biases initial values
weights_W = cat(3,W,W,W);
```

```
bias_w0 = cat(3,w0,w0,w0);
weights_Z = cat(3,Z,Z,Z);
bias_z0 = cat(3,z0,z0,z0);

  for i = 3:length(r)-1
    % process output
    y(i) = 0.7*y(i-1)+0.3*u(i-1)+0.05*u(i-2);

    % future target value normalisation
    r_N = a_y*r(i+1)+b_y;

    % actual output value normalisation
    y_N(i) = a_y*y(i)+b_y;
    % stimulus n° i
    x = [r_N y_N(i) y_N(i-1) y_N(i-2) u_N(i-1) u_N(i-2)]';

    % hidden layer's cells activation
    h_in = W*x+w0;
    h_in = h_in.*(-5<=h_in<=5)-5.*(h_in<-5)+5.*(h_in>5);

    % hidden layer's cells output
    h = logsig(h_in);

    % output layer's cells activation
    o_in = Z*h+z0;
    o_in = o_in.*(-5<=o_in<=5)-5.*(o_in<-5)+5.*(o_in>5);

    % output layer response
    o = logsig(o_in);

    % normalised control signal
    u_N(i) = o;

    % control signal de-normalisation
    u(i) = (u_N(i)-b_u)/a_u;

    % weights and biases updating
    e = r(i)-y(i);

    % error to be back-propagate
    delta = o.*(ones(1,1)-o).*e;

    % weights Z and bias z0 updating
    Z = Z+eta*delta*h';
    z0 = z0+eta*delta;

    % error in output hidden layer
    dh = h.*(ones(size(h))-h).*(Z'*delta);

    % weights W and bias w0 updating
    W = W+eta*dh*x';
```

```
    w0 = w0+eta*db;

    % saving weights and bias
    weights_W = cat(3,weights_W,W);
    bias_w0 = cat(3,bias_w0,w0);
    weights_Z = cat(3,weights_Z,Z);
    bias_z0 = cat(3,bias_z0,z0);

  end % for loop

% plotting the different signals
figure(1)
plot(y)
hold on
plot(r,'-.')
title(['target and output signals; Gain = ' num2str(eta)])
xlabel('discrete time'), grid

figure(2), stairs(u)
title(['control signal, Gain = ' num2str(eta)])
xlabel('discrete time'), grid

% plotting the weights W11, W12, W13 and W14 evolution
figure(3)
for k = 1:5
   x = weights_W(k,1,:);
   plot(x(:)), hold on
end
hold off, grid
title(['weights W1k adaptation, Gain = ' num2str(eta)])
xlabel('discrete time')

% plotting hidden biases evolution
figure(4)
for k = 1:5
   x = bias_w0(k,1,:);
   plot(x(:)), hold on
end
hold off, grid
title(['hidden layer bias , Gain = ' num2str(eta)])
xlabel('discrete time')

% plotting the weights Z evolution by adaptation
figure(5)
for k = 1:5
   x = weights_Z(1,k,:);
   plot(x(:)), hold on
end

hold off, grid
title(['weights Z adaptation, Gain = ' num2str(eta)])
```

```
xlabel('discrete time')

% plotting the bias z0 evolution by adaptation
figure(6), plot(bias_z0(:)), grid
title(['output cell bias, Gain = ' num2str(eta)])
xlabel('discrete time')
```

在自适应增益等于 0.1 时，响应时间足够长，所以输出信号和目标信号有一定差别（见图 4-62）。

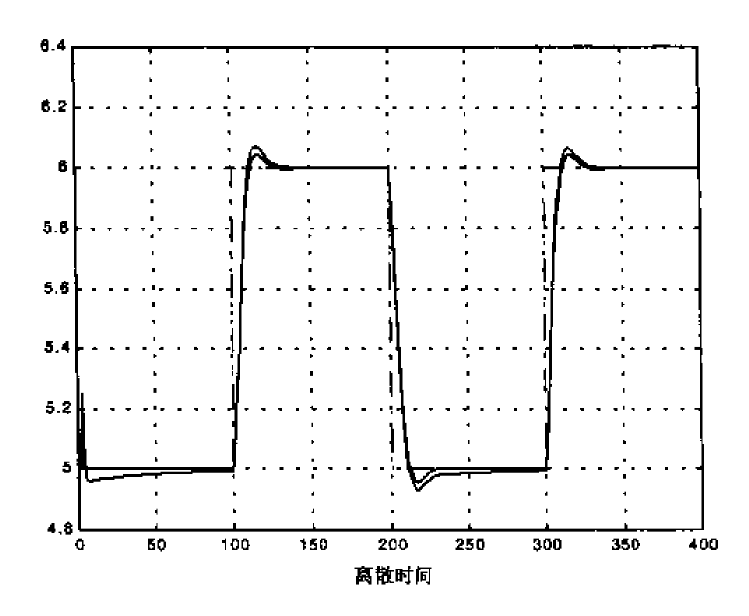


图 4-62　自适应增益等于 0.1 时的目标信号和输出信号

较大的自适应增益值可能缩短响应时间并提高跟踪质量。图 4-63 为自适应增益等于 0.5 时的目标信号和输出信号。

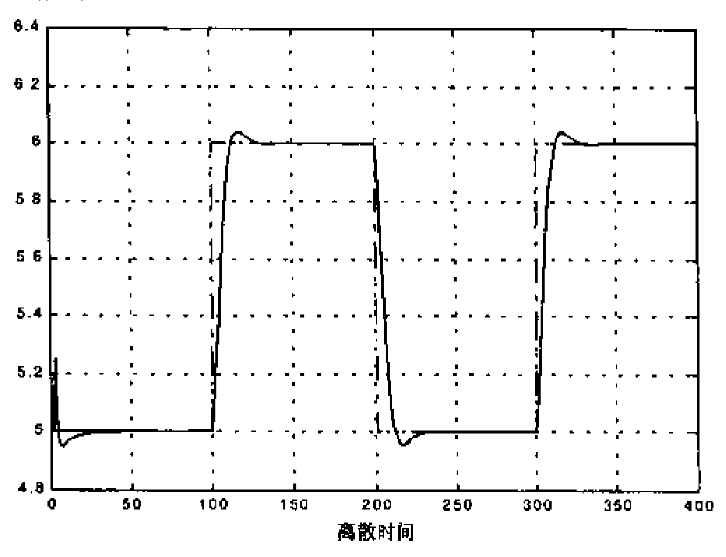


图 4-63　自适应增益等于 0.5 时的目标信号和输出信号

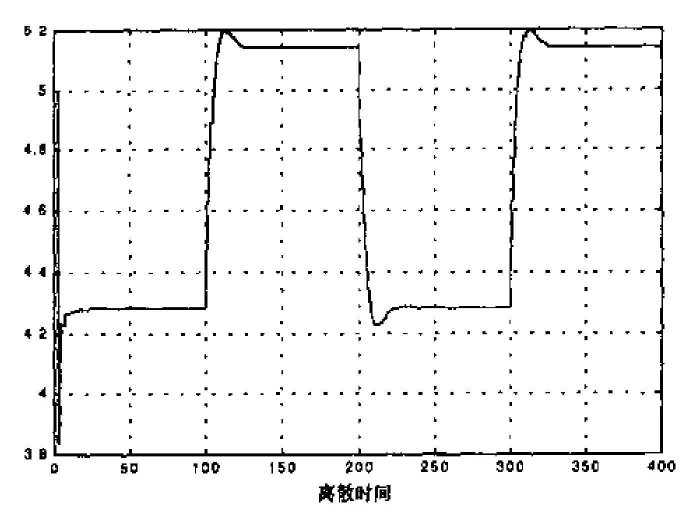


图 4-64　自适应增益等于 0.5 时的控制信号

可以看到，与它们在学习阶段得到的值相比，连接输入层和隐含层的权值没有变化，连接隐含层和输出节点的权值也同样没有变化。相反，偏移量根据跟踪的目标信号发生了变化（见图 4-65、图 4-66）。

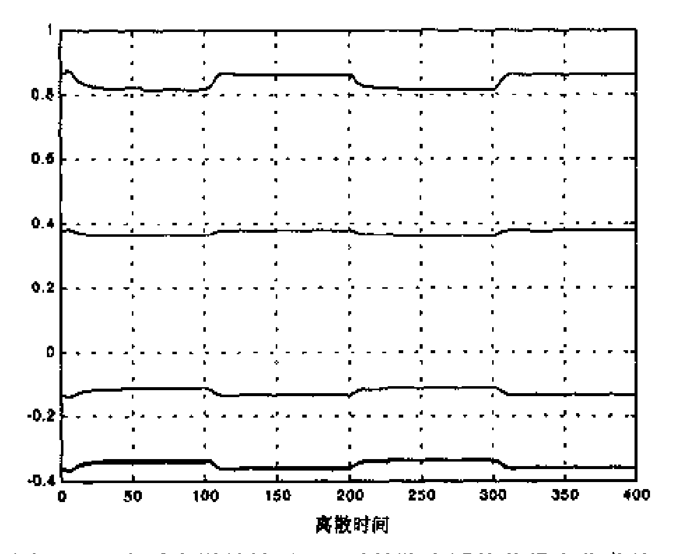


图 4-65　自适应增益等于 0.5 时的隐含层偏移量变化曲线

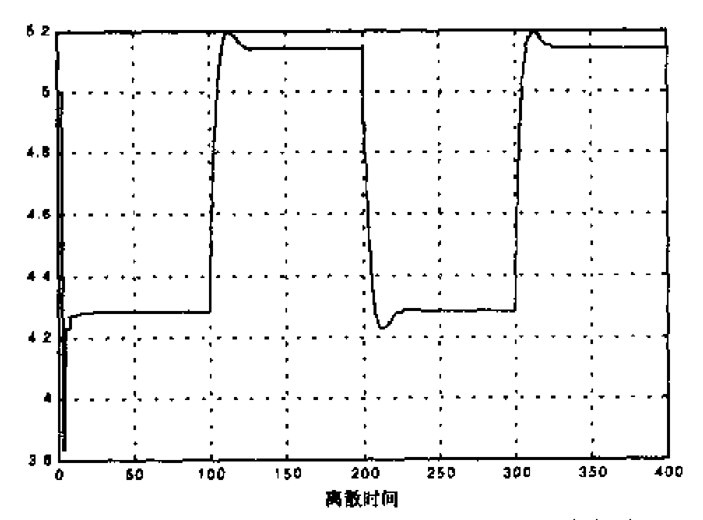


图 4-66　自适应增益等于 0.5 时的输出层偏移量变化曲线

4.5 信号预测

我们采用一个神经网络对每个采样间隔的信号值进行预测。对图 4-67 所示的神经网络结构进行实时学习，以估计预测信号的变化。

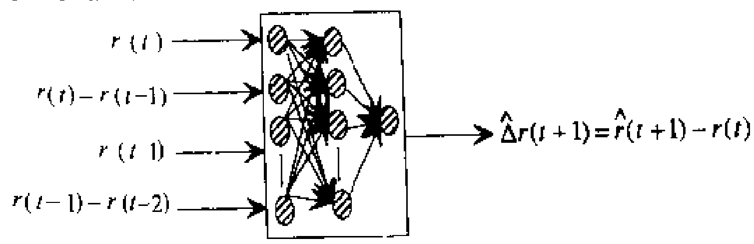


图 4-67　预测信号变化的神经网络结构

通过在每个离散时刻 t 增加有效值，可以推导出估计预测值，如图 4-68 所示。在 t 时刻和（t-1）时刻之间信号的变化误差在神经网络中反向传播以便进行学习。

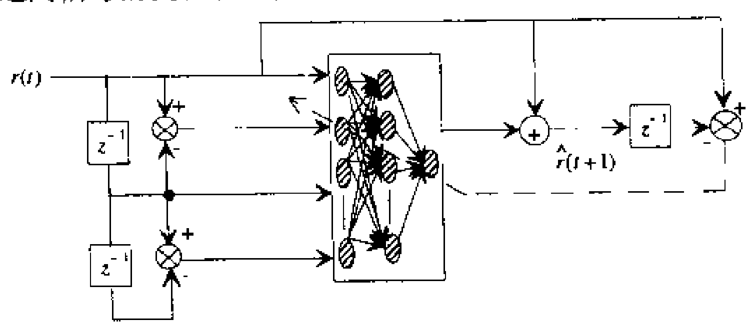


图 4-68　信号预测系统结构框图

pred.m

```
% One step ahead signal prediction
clear all; close all; clc

% input cells (4), hidden cells (5) and output cell (1)
Nb_Iu = 4; Nb_Hu = 5; Nb_Ou = 1;

% Randomly initialisation of weights W, Z and biases w0 and z0
W = rand(Nb_Hu,Nb_Iu);
w0 = rand(Nb_Hu,1);

Z = rand(Nb_Ou,Nb_Hu);
z0 = rand(Nb_Ou,1);

% signal to be predicted
t = 0:0.01:3;
r = 0.5*abs(sin(5*t).*exp(-.5*t)+cos(2*t)) ;

r_est = r;

% weights and bias initial values
weights_W = cat(3,W,W,W);
```

```
bias_w0 = cat(3,w0,w0,w0);
weights_Z = cat(3,Z,Z,Z);
bias_z0 = cat(3,z0,z0,z0);

eta = 20; % learning gain
  for i = 3:length(r)-1

    % stimulus n° i
    x = [r(i) r(i)-r(i-1) r(i-1) r(i-1)-r(i-2)]';

    % estimation of the future variation future
    h = logsig(W*x+w0);
    o = logsig(Z*h+z0);
    dr_est(i+1) = o;

    % future estimated signal value
    r_est(i+1) = dr_est(i+1)+r(i);

    % estimation error
    e(i) = r(i)-r(i-1)-dr_est(i);

    % error to back-dropropagate
    delta = o.*(ones(1,1)-o).*e(i);

    % updating weights Z and bias z0 matrices
    Z = Z+eta*delta*h';
    z0 = z0+eta*delta;

    % error in hidden layer
    dh = h.*(ones(size(h))-h).*(Z'*delta);

    % updating weights W and bias w0 matrices
    W = W+eta*dh*x';
    w0 = w0+eta*dh;

    % saving weights and biases
    weights_W = cat(3,weights_W,W);
      bias_w0 = cat(3,bias_w0,w0);
      weights_Z = cat(3,weights_Z,Z);
    bias_z0 = cat(3,bias_z0,z0);
  end % for loop

% variation of the estimation error
 e = e(250:300);
 disp('estimation''s error variance');
 var_err = std(e)^2

% plotting of different signals
figure(1), hold on
h = plot(r_est);
set(h,'LineWidth',2), plot(r)
```

```
axis([0 length(r) 0 1.7]), grid
title(['real and predicted signals, Gain = ' num2str(eta)])

figure(2)
plot(r-r_est), hold off
title('prediction error')
axis([0 length(t) -0.1 0.1]), grid

% weights W11, W12, W13 and W14 evolution plot
figure(3)
for k = 1:Nb_Hu
   x = weights_W(k,1,:);
   plot(x(:)), hold on
end
hold off, grid
title(['weights W1k evolution , Gain = ' num2str(eta)])
xlabel('discrete time')

% weights Z evolution's plotting
figure(5)
for k = 1:Nb_Hu
   x = weights_Z(1,k,:);
   plot(x(:)), hold on
end
hold off, grid
title(['weights Z evolution, Gain = ' num2str(eta)])
xlabel('discrete time')
```

图 4-69 显示了实际信号和神经网络的一步预测值。

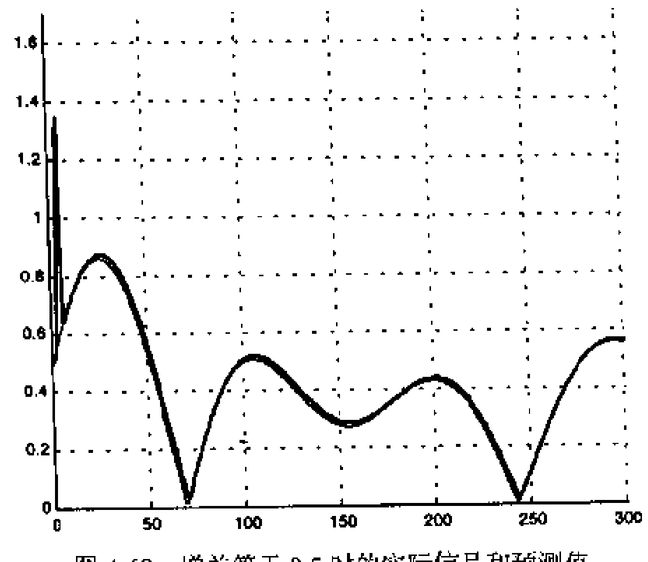


图 4-69　增益等于 0.5 时的实际信号和预测值

```
estimation's error variance
var_err =
  2.1382e-005
```

如果希望神经网络直接产生预测值的估计值，则必须把 t 时刻的有效信号值和网络输出之间的差反向传播。

文件 pred2.m 实现了这类估计。

```
% network output : estimate of  r(t+1)
o = logsig (z*h+z0);
r_est(i +1) = o

% estimation error
e (i) = r(i) -r_est (i);
```

增益等于 4 时不仅能够正确估计信号，而且会出现初始不稳（见图 4-70）。

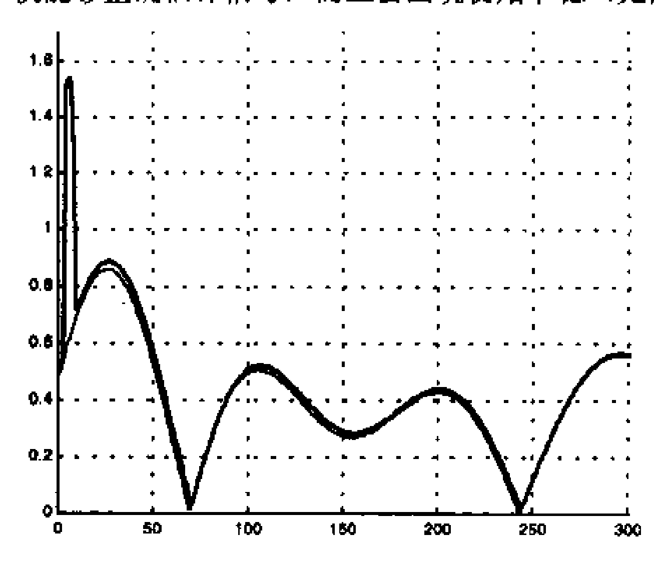


图 4-70　增益等于 4 时的实际信号和预测值

```
estimation's error variance
var_err =
  2.0598e-005
```

当增益等于 5 时不稳定性很明显（见图 4-71）。

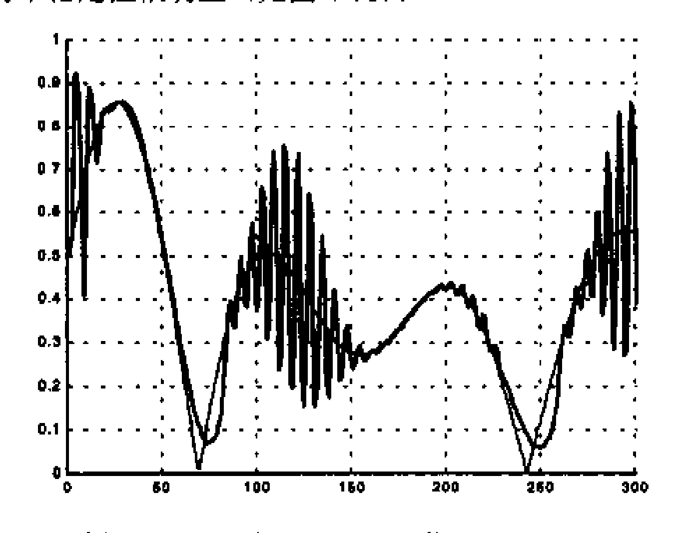


图 4-71　增益等于 5 时的实际信号和预测值

当神经网络输出变化时预测效果更好，因为 t 时刻有效的信号值被引入并增加到估计变化量。

如果神经网络采用第二个信号的导数，即如下式所示的响应，就可以提高预测质量：

$$\hat{r}(t+2)-2\hat{r}(t+1)+r(t)$$

为得到预测估计值，可以应用两个信号真值，把 $2r(t)-r(t-1)$ 增加到前一个时刻的神经网络估计值（见图 4-72）。

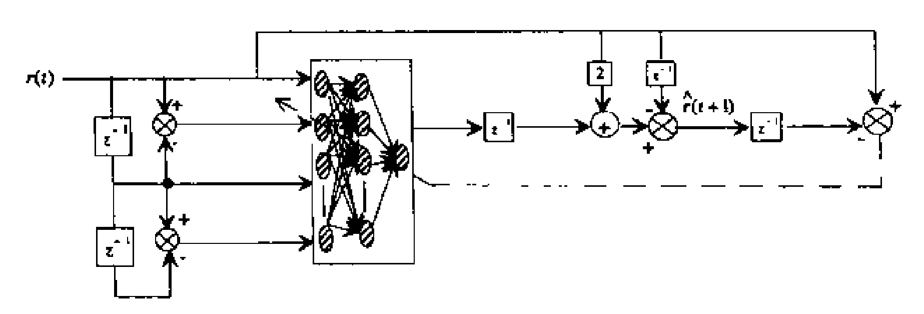


图 4-72　系统结构框图

文件 pred3.m 的命令行不同于文件 pred.m 的命令行。

```
% future estimated variation by the network
h = logsig(W*x +w0); o = logsig (Z*h +z0); ddr_est(i +1) = o;
% signal future estimated value
r_est (i+1) = ddr_est(i+1) +2*r(i)-r(i-1);
e(i) = r(i) -2*r(i -1)-ddr_est (i) ; %estimation error
```

在这个例子中，学习增益可能达到 10，它能够提高估计质量且保证稳定性（见图 4-73）。

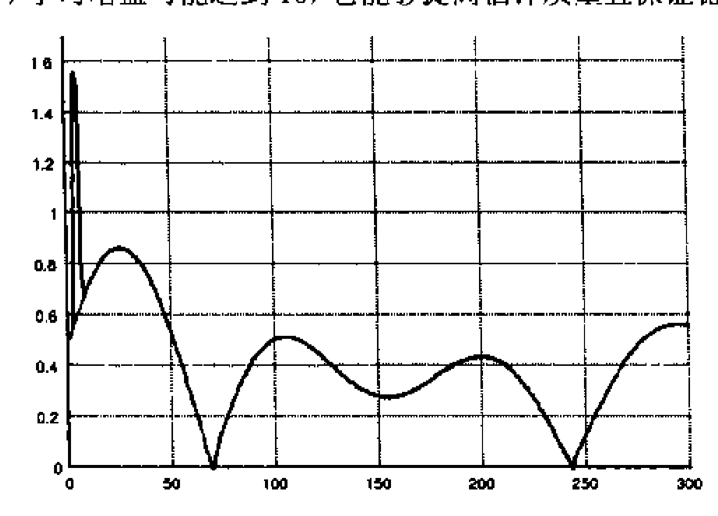


图 4-73　增益等于 10 时的实际信号和预测值

估计误差的方差接近于 10^{-8}，等于简单学习变化量时的 1/2000。

第5章　自适应滤波

自适应滤波是随机信号处理的一个重要部分。数字信号处理器的发展使得它的实现变得简单，并且能作用于宽带频谱的快速信号。

自适应滤波的主要优点是能够消除特性随时间变化的噪声，这在系数固定的非自适应结构中无法做到。此外，在信号预测和过程辨识中也会用到它。

5.1　自适应滤波原理

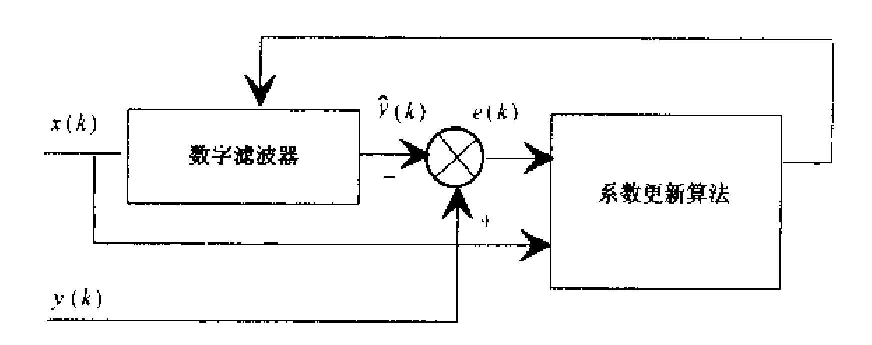


图5-1　自适应滤波原理图

用误差信号 $e(k)$和输入量 $x(k)$更新滤波器系数（见图5-1）。

自适应滤波器可根据以下几点进行分类：

- 计算系数的算法；
- 优化性能指标；
- 数字滤波器的结构。

大多数情况下，最优化准则和算法是相关联的。最常见的准则和算法如下：

- 利用梯度算法的最小均方准则；
- 利用符号算法的最小绝对值准则；
- 利用递推算法的最小二乘准则。

能运用的最简单的算法是梯度法。

理论

记数字滤波器脉冲响应为：

$$h(k)=[h_0(k)\quad h_1(k)\quad \cdots \quad h_{n-1}(k)]^{\mathrm{T}}$$

输入采样信号为：

$$x(k)=[x(k)\quad x(k-1)\quad \cdots \quad x(k-n+1)]$$

误差信号为：

$$e(k)=y(k)-\hat{y}(k)$$

$$e(k)=y(k)-h^{\mathrm{T}}(k)x(k)$$

优化过程就是最小化性能指标 $J(k)$，它是误差的平方和：

$$J(k)=\sum_{i=1}^{k}\left[y(i)-h^{\mathrm{T}}(k)x(i)\right]^2$$

然后，求使 $J(k)$最小的系数向量 $h(k)$，即使 $J(k)$对 $h(k)$的导数为零，也就是$\frac{\mathrm{d}J(k)}{\mathrm{d}h(k)}=0$。把 $J(k)$的表达式代入，得：

$$2\sum_{i=1}^{k}\left[y(i)-h^{\mathrm{T}}(k)x(i)\right]x(i)=0$$

和

$$\sum_{i=1}^{k}x^{\mathrm{T}}(i)y(i)=h^{\mathrm{T}}(k)\sum_{i=1}^{k}x(i)x^{\mathrm{T}}(i)$$

由此得出滤波器系数的最优向量：

$$h^{\mathrm{T}}(k)=\frac{\sum_{i=1}^{k}x^{\mathrm{T}}(i)y(i)}{\sum_{i=1}^{k}x(i)x^{\mathrm{T}}(i)}$$

这个表达式由输入信号自相关矩阵 $C_{xx}(k)$和输入信号与参考信号的相关矩阵 $C_{yx}(k)$组成，如下所示，维数都为(n, n)：

$$C_{xx}(k)=\sum_{i=1}^{k}x(i)x^{\mathrm{T}}(i),\qquad C_{yx}(k)=\sum_{i=1}^{k}x^{\mathrm{T}}(i)y(i)$$

系数最优向量也可以写成如下形式：

$$h^{\mathrm{T}}(k)_{\mathrm{opt}}=C_{yx}(k)C_{xx}^{-1}(k)$$

推导一个系数的递归表达式以便能从 $h(k-1)$向量得到 $h(k)$，而 $h(k-1)$也由前一时刻的值得到。

首先写出自相关和互相关矩阵的递归表达式：

$$C_{xx}(k)=C_{xx}(k-1)+x(k)x^{\mathrm{T}}(k)$$
$$C_{yx}(k)=C_{yx}(k-1)+x^{\mathrm{T}}(k)y(k)$$

把 $C_{yx}(k)$的递归表达式代入系数向量表达式，得：

$$h^{\mathrm{T}}(k)=C_{yx}(k)C_{xx}^{-1}(k)$$

即

$$h^{\mathrm{T}}(k)=[C_{yx}(k-1)+x^{\mathrm{T}}(k)y(k)]C_{xx}^{-1}(k)$$

考虑到

$$C_{yx}(k-1)=h^{\mathrm{T}}(k-1)C_{xx}(k-1)$$

可以记

$$h(k)=C_{xx}^{-1}(k)\left[C_{xx}(k-1)h(k-1)+y(k)x(k)\right]$$

用前面得到的表达式求出$C_{xx}(k-1)$，并代入上式：

$$h(k)=C_{xx}^{-1}(k)\left\{[C_{xx}(k)-x(k)x^{\mathrm{T}}(k)]\,h(k-1)+y(k)x(k)\right\}$$

或

$$h(k)=h(k-1)+C_{xx}^{-1}(k)\left[y(k)x(k)-x(k)x^{\mathrm{T}}(k)h(k-1)\right]$$

则滤波器系数的递归关系式可以记作

$$h(k)=h(k-1)+C_{xx}^{-1}(k)\left[y(k)x(k)-x(k)x^{\mathrm{T}}(k)h(k-1)\right]$$

其中

$$y(k)-x^{\mathrm{T}}(k)h(k-1)=e(k)$$

$e(k)$表示先验误差，因为它是由前一个采样时刻的系数算出的。在实际中，很多时候由于 $h(k)$ 计算的复杂性而不能应用于实时控制。用δ、I 代换 $C_{xx}(k)$，其中：

- δ　自适应梯度；
- I　辨识矩阵(n, n)。

这时

$$h(k)=h(k-1)+\delta\, x(k)e(k)$$

那么它就不再是一个严格最小二乘准则的问题，而是一个最小均方准则的问题。

5.2　梯度算法，LMS 准则

梯度算法的方程表示如下：

$$\begin{cases} e(k)=y(k)-x^{\mathrm{T}}(k)h(k-1) \\ h(k)=h(k-1)+\delta x(k)e(k) \end{cases}$$

5.2.1　自适应梯度δ的选择

如果后验误差小于或等于先验误差，就认为算法稳定，即

$$\varepsilon(k)=h(k)-h_{\mathrm{opt}}$$

$$\varepsilon(k+1)=h(k+1)-h_{\mathrm{opt}}$$

如果系统稳定：

$$E\left[\varepsilon(k+1)\right]<E\left[\varepsilon(k)\right]$$

其中：

$$\varepsilon(k+1)=h(k)+\delta x(k+1)e(k+1)-h_{\mathrm{opt}}$$

$$\varepsilon(k+1)=h(k)+\delta x(k+1)[y(k+1)-h^{\mathrm{T}}(k)x(k+1)]-h_{\mathrm{opt}}$$

计算得：

$$\varepsilon(k+1)=\varepsilon(k)-\delta x(k+1)x^{\mathrm{T}}(k+1)\varepsilon(k)$$

所以，满足 $E[\varepsilon(k+1)]<E[\varepsilon(k)]$ 的条件可写为：

$$\left|E\left[1-\delta x^2(k+1)\right]\right|<1$$

对于一个均值为零的信号：

$$E\left[x^2(k+1)\right]=n\sigma_x^2$$

由此得用自适应梯度表示的稳定性条件：

$$0<\delta<\frac{2}{n\sigma_x^2}$$

则有效的δ选择由自适应速度与残差的折衷产生，δ越大越好，并且残差和δ成比例。

5.2.2　自适应速度，滤波器时间常数

时间常数最小时，滤波器收敛最快，其关系式如下：

$$\tau_{\min} = nT_s$$

其中：

- n　滤波器系数；
- T_s　采样周期。

<u>注意</u>　之所以称为梯度算法是因为误差平方的梯度与 $x(k)e(k)$成比例。事实上，

$$e^2(k) = y^2(k) - 2x^T(k)h(k-1)y(k) + \left[x^T(k)h(k-1)\right]^2$$

即

$$\frac{\partial e^2(k)}{\partial h(k-1)} = -2x^T(k)\left[y(k) - x^T(k)h(k-1)\right]$$

$$\frac{\partial e^2(k)}{\partial h(k-1)} = -2x^T(k)e(k)$$

5.3　递推最小二乘算法，严格最小二乘算法

最小二乘算法和 LMS 算法的主要不同之处在于它用到的信息包括前一时刻输入信号的采样。与同阶数的 LMS 自适应滤波器相比，它的收敛时间更短，但这个性能的提高是以增加复杂性和计算时间为代价的。

仍以先前采用的系统方程为例：

$$\begin{cases} e(k+1) = y(k+1) - h^T(k)x(k+1) \\ h(k) = h(k-1) + C_{xx}^{-1}x(k)\left[y(k) - x^T(k)h(k-1)\right] \end{cases}$$

协方差矩阵 $C_{xx}^{-1}(k)$ 可由下面的逆变换定理表示成递归形式：

$$\begin{cases} A = B + CDC^T \\ A^{-1} = B^{-1} - B^{-1}C\left[C^TB^{-1}C + D^{-1}\right]^{-1}C^TB^{-1} \end{cases}$$

即

$$C_{xx}(k) = C_{xx}(k-1) + x(k)x^T(k)$$

由此类推

$$\begin{cases} B = C_{xx}(k-1) \\ C = x(k) \\ D = I \end{cases}$$

得到

$$C_{xx}^{-1}(k) = C_{xx}^{-1}(k-1) - \frac{C_{xx}^{-1}(k-1)x(k)x^T(k)C_{xx}^{-1}(k-1)}{1 + x^T(k)C_{xx}^{-1}(k-1)x(k)}$$

定义自适应增益

$$g(k) = \frac{C_{xx}^{-1}(k-1)x(k)}{1 + x^T(k)C_{xx}^{-1}(k-1)x(k)}$$

得到

$$C_{xx}^{-1}(k)=C_{xx}^{-1}(k-1)-g(k)x^{\mathrm{T}}(k)C_{xx}^{-1}(k-1)$$

系数也能够以自适应增益 $g(k)$的形式表示：

$$h^{\mathrm{T}}(k)=C_{yx}(k)C_{xx}^{-1}(k)$$

即

$$h^{\mathrm{T}}(k)=C_{yx}(k-1)C_{xx}^{-1}(k)+x^{\mathrm{T}}(k)y(k)C_{xx}^{-1}(k)$$

把 $C_{xx}^{-1}(k)$ 的表达式代入上式：

$$h^{\mathrm{T}}(k)=C_{yx}(k-1)C_{xx}^{-1}(k)-C_{yx}(k-1)C_{xx}^{-1}(k)g(k)x^{\mathrm{T}}(k)+x^{\mathrm{T}}(k)y(k)C_{xx}^{-1}(k)$$

$$h^{\mathrm{T}}(k)=h^{\mathrm{T}}(k-1)-h^{\mathrm{T}}(k-1)g(k)x^{\mathrm{T}}(k)+x^{\mathrm{T}}(k)y(k)C_{xx}^{-1}(k)$$

自适应增益的表达式也可写成如下形式：

$$g(k)+g(k)x^{\mathrm{T}}(k)C_{xx}^{-1}(k-1)x(k)=C_{xx}^{-1}(k-1)x(k)$$

即

$$g(k)=C_{xx}^{-1}(k-1)x(k)$$

把上式代入 $h^{\mathrm{T}}(k)$表达式得到

$$h^{\mathrm{T}}(k)=h^{\mathrm{T}}(k-1)-h^{\mathrm{T}}(k-1)g(k)x^{\mathrm{T}}(k)+g(k)y(k)$$

即

$$h(k)=h(k-1)+g(k)[y(k)-h^{\mathrm{T}}(k-1)x(k)]$$

所以递推最小二乘算法可总结成如下方程式：

$$\begin{cases}g(k)=\dfrac{C_{xx}^{-1}(k-1)x(k)}{1+x^{\mathrm{T}}(k)C_{xx}^{-1}(k-1)x(k)}\\ C_{xx}^{-1}(k)=C_{xx}^{-1}(k-1)-g(k)x^{\mathrm{T}}(k)C_{xx}^{-1}(k-1)\\ h(k)=h(k-1)+g(k)[y(k)-h^{\mathrm{T}}(k-1)x(k)]\\ e(k)=y(k)-h^{\mathrm{T}}(k-1)x(k)\end{cases}$$

应用

设初始化C_{xx}^{-1}=kI，其中 I 是单位矩阵（n、n），k 是一个保证算法快速收敛的系数。如果没有任何滤波器系数的先验知识（在实际中通常都是这种情况），那么 $h(k)$=0。现在，只需计算每个采样周期的：

- 自适应增益 $$g(k)=\frac{C_{xx}^{-1}(k-1)x(k)}{1+x^{\mathrm{T}}(k)C_{xx}^{-1}(k-1)x(k)}$$
- 先验误差 $$e(k)=y(k)-h^{\mathrm{T}}(k-1)x(k)$$
- 系数向量 $$h(k)=h(k-1)+g(k)e(k)$$
- 自相关矩阵 $$C_{xx}^{-1}(k)=C_{xx}^{-1}(k-1)-g(k)x^{\mathrm{T}}(k)C_{xx}^{-1}(k-1)$$

注意

前面的算法只适用于固定的信号。由于自适应增益很快减小到较小值，使得滤波器不能再根据输入信号的统计变化有效地调节。对于不固定信号的自适应控制，可以采用其他增益可变的方法，这些方法引入了一个称为遗忘因子的辅助因子 λ。这个系数之所以叫做遗忘因子，是因为它能影响算法的“记忆”。在信号不固定时，可以根据输入的统计变化来选择它：

$$0<\lambda<1$$

修正的最小二乘递推算法可以写为如下形式，可计算每个采样周期的以下参数：

- 自适应增益 $g(k)=\dfrac{\lambda^{-1}C_{xx}^{-1}(k-1)x(k)}{1+x^{\mathrm{T}}(k)C_{xx}^{-1}(k-1)x(k)}$

- 先验误差

$$e(k)=y(k)-h^{\mathrm{T}}(k-1)x(k)$$

- 系数向量

$$h(k)=h(k-1)+g(k)e(k)$$

- 自相关矩阵

$$C_{xx}^{-1}(k)=\lambda^{-1}C_{xx}^{-1}(k-1)-\lambda^{-1}g(k)x^{\mathrm{T}}(k)C_{xx}^{-1}(k-1)$$

遗忘因子的选择

◆ 遗忘因子固定

$$\lambda(k)=\lambda=\mathrm{cte}$$

一般选择

$$\lambda\text{=0.9à0.99}$$

最小性能指标

$$J(k)=\sum_{i=0}^{n-1}\lambda^{n-i}\left|e(i)\right|^2$$

随着采样时刻的变化，系数逐渐减小，适用于输入信号慢变过程。

◆ 遗忘因子可变

$$\lambda(k)=K\lambda(k-1)+1-K$$

一般选择

$$K\text{=0.9à0.99，}\lambda(1)\text{=0.9à0.99}$$

最小性能指标

$$J(k)=\sum_{i=0}^{n-1}\lambda^{n-i}\left|e(i)\right|^2$$

k 取值越大，$\lambda(k)$ 越趋向于 1。

这种方法适用于固定信号并产生一个加速收敛过程以避免自适应增益的衰减太快。

◆ 遗忘因子保持固定轨迹

目的是调整每步采样的 $\lambda(k)$，以保持 $C_{xx}^{-1}(k)$ 矩阵为固定轨迹，这样避免了增益趋向于零。这种方法适用于输入信号有很大变化的情况。$\lambda(k)$可由下式计算：

$$\mathrm{trace}\left[C_{xx}^{-1}(k)\right]=\frac{1}{\lambda(k)}trace\left[C_{xx}^{-1}(k-1)-\frac{C_{xx}^{-1}(k-1)x(k)x^{T}(k)C_{xx}^{-1}(k-1)}{1+x^{T}(k)C_{xx}^{-1}(k-1)x(k)}\right]$$

5.4 LMS 自适应滤波器应用举例

5.4.1 自回归过程的自适应预估器

自回归过程是用来描述伴随一些可能性规律出现的统计现象的瞬时估计的随机过程。

◆ 自回归模型

一个 n 阶自回归模型如图 5-2 所示。

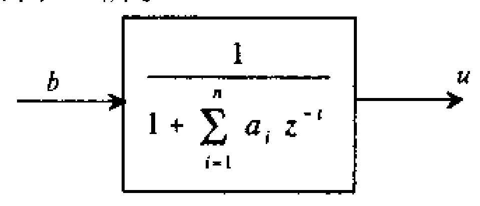


图 5-2 n 阶自回归模型

联系输入 b 和输出 u 循环的方程如下：

$$u(k)+a_1u(k-1)+a_2u(k-2)+\cdots+a_nu(k-n)=b(k)$$

其中 $b(k)$是零均值白噪声，可以写成如下形式：

$$u(k)=-\sum_{i=1}^{n}a_iu(k-i)+b(k)$$

$u(k)$等于 k 时刻以前的控制量的线性组合加上一个噪声或误差项。这个过程的 Z 变换如下：

$$u(k)+a_1u(k)z^{-1}+a_2u(k)z^{-2}+\cdots+a_nu(k)z^{-n}=b(k)$$

$$T(z)=\frac{b(z)}{u(z)}=1+\sum_{i=1}^{n}a_iz^{-i}$$

这个过程式的两个应用举例如下。

● 已知输入序列 $u(k)$，用滤波器产生白噪声（见图 5-3）。

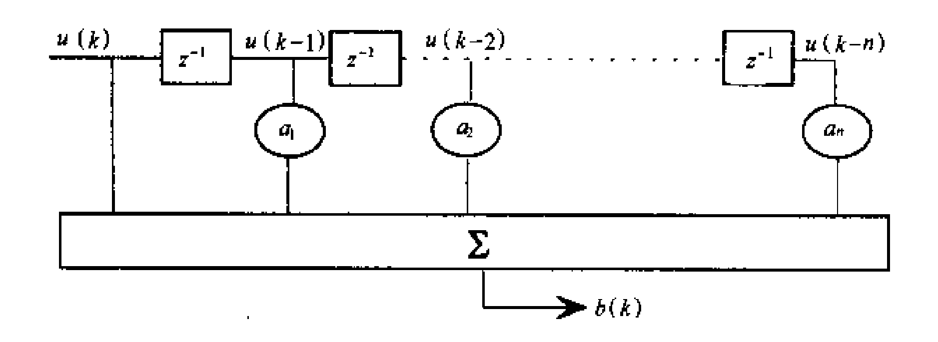


图 5-3 应用一模型

● 已知白噪声，用滤波器产生输入序列 $u(k)$，得到一个传递函数过程如下：

$$H(z)=\frac{u(z)}{b(z)}=\frac{1}{T(z)}$$

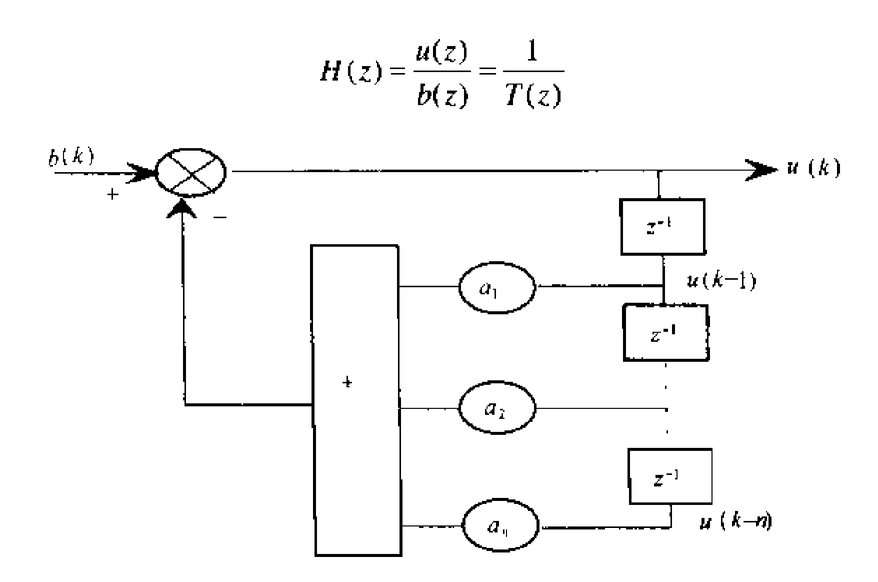


图 5-4 应用二模型

为了应用 LMS 算法，应用如下一阶自回归模型：

$$y(k)=-a_1 y(k-1)+b(k)$$

a_1是模型的惟一参数，$b(k)$是均值为零的白噪声。用一个自适应滤波器生成一个可以对参数a_1进行一步预测的一阶自适应预估器。LMS 算法可由如下方程表示：

$$e(k)=y(k)-\hat{a}_1 y(k-1)$$

$$\hat{a}_1(k+1)=\hat{a}_1(k)+\delta y(k-1)e(k)$$

取 N 个点估计参数 a_1，为获取平均值重复 M 次，而且分别对δ=0.01，δ=0.05，δ=0.1进行计算。参数a_1固定在-0.8。文件 predictor_lms.m 应用了这个梯度算法预估器。

predictor_lms.m file

```
% Adaptive predictor for AR process
N = 500;                % Iterations number
M = 20;                 % Number of tests for average
n = 1;                  % Predictor order
a1 = -0.8;              % Predictor coefficient
h = zeros(M,n+1,3);     % Estimated coefficients matrix
e = zeros(M,n,3);       % Estimation errors matrix

% Calculation loop for delta = 0.01, 0.05 and 0.1
for d = 1:3,
  if d==1 delta = 0.01;
  else delta = 0.05*(d-1);
  end;
  % M tests loop
  for k = 1:M,
  b = 0.2*randn(1,N);
  y(1) = 1;
    % N iterations calculation loop of the input sequence
    for i = 2:N,
      y(i) = -a1*y(i-1)+b(i);
    end
```

```
    % N iterations calculation loop
  % of the adaptive predictor
    for i = n+1:N,
      e(k,i,d) = y(i)-h(k,i,d)*y(i-1);
      h(k,i+1,d) = h(k,i,d)+delta*y(i-1)*e(k,i,d);
    end
  end
end

% tests averages calculation
for d = 1:3,
  for i = 1:N,
    em(i,d) = 0;
    hm(i,d) = 0;
    for j = 1:M
      em(i,d) = em(i,d)+e(j,i,d)^2;
      hm(i,d) = hm(i,d)+h(j,i,d);
    end
   em(i,d) = em(i,d)/M;
   hm(i,d) = hm(i,d)/M;
  end
end

% Results graphs
figure(1)
semilogy(1:150,em(1:150,1)), hold on
semilogy(1:150,em(1:150,2),'r:'), hold on
semilogy(1:150,em(1:150,3),'g.'), hold off
axis([0 150 1e-2 1]), grid
title('Mean square error E[e^2(t)]')
xlabel('Samples')
gtext('\leftarrowd = 0.01'), gtext('\leftarrowd = 0.05')
gtext('\leftarrowd = 0.1')
figure(2), plot(1:N,hm(1:N,1)), hold on
plot(1:N,hm(1:N,2),'r'), hold on
plot(1:N,hm(1:N,3),'g'), hold off, grid
title('Filter coefficient evolutions')
xlabel('Samples'),
gtext('d=0.01'), gtext('d=0.05'), gtext('d=0.1')
```

δ越大，误差信号衰减越快，但它的变化更重要。在大约20次振荡后，误差小于0.1。

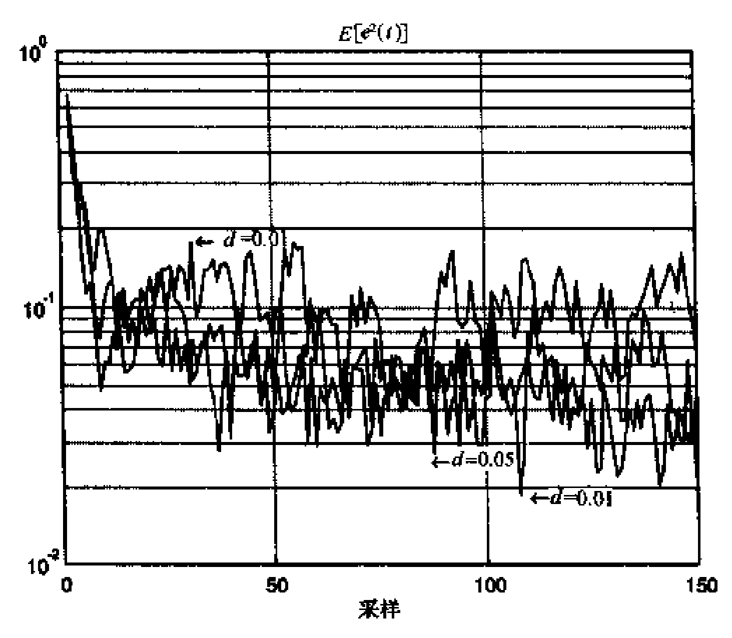


图 5-5　平均方差 $E[e^2(t)]$曲线

系数以时间常数的指数曲线收数，δ 越大，时间常数越小。

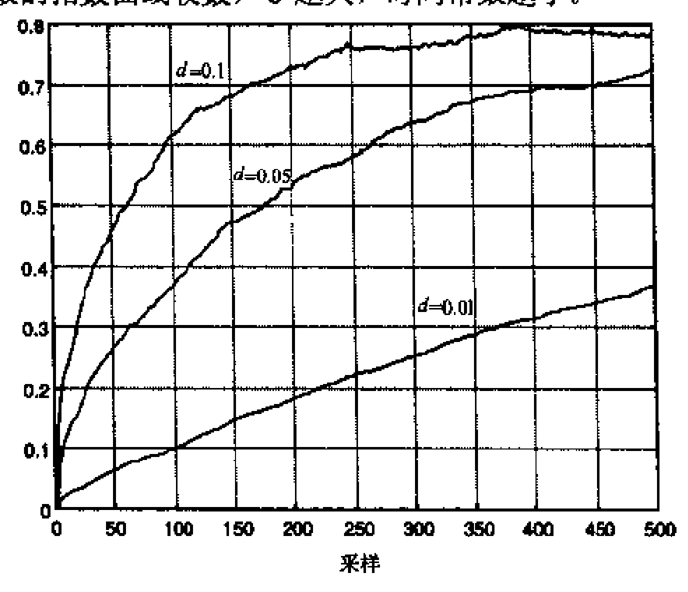


图 5-6　滤波器系数曲线

若 $0<\delta<\dfrac{2}{n\sigma_x^2}$，自适应算法稳定，其中 n=1，σ_x^2=输入量方差，则当方差等于 0.0390 时，$\delta_{\max}\approx 50$。

5.4.2　消除干扰

设 $s(k)$为输入信息，$b_0(k)$是与它弱相关的噪声。$x_r(k)$是和 $b_0(k)$密切相关的参考信号：

$$x(k)=s(k)+b_0\cos(\omega_0 k+\Phi_0)$$

$$x_r(k)=b\cos(\omega_0 k+\Phi_0)$$

$x_r(k)$用来产生一个干扰图像，把干扰从 $x(k)$中提取出来，这样信噪比就大大地提高了。这

个自适应滤波器为中心频率等于 $\frac{\omega_0}{2\pi}$ 的抑制滤波器。自适应系数的选择可以用来调整滤波器的通频特性（见图 5-7）。

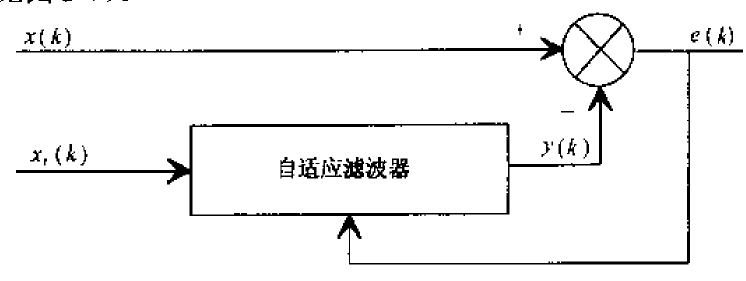


图 5-7　系统结构框图

滤波器的计算方法可由下面的方程表示：

$$y(k)=\sum_{i=0}^{n-1}\hat{a}_i(k)x_r(k-i)$$

$$e(k)=x(k)-y(k)$$

$$\hat{a}_i(k+1)=\hat{a}_i(k)+\delta x_r(k-i)e(k)$$

为了求出综合传递函数，需采用 Z 变换。设 $T(z)=\frac{y(z)}{e(z)}$，则系统可如下表示：

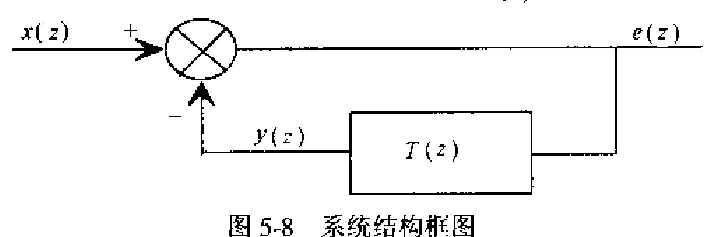


图 5-8　系统结构框图

$$TZ\left[x_r(k-i)e(k)\right]=TZ\left[b\cos(\omega_0(k-i)+\Phi)e(k)\right]$$

把 $\Phi_i=\Phi-\omega_0 i$ 代入得：

$$TZ\left[x_r(k-i)e(k)\right]=TZ\left[\frac{b}{2}\left(\mathrm{e}^{\mathrm{j}(\omega_0 k+\Phi_i)}+\mathrm{e}^{-\mathrm{j}(\omega_0 k+\Phi_i)}\right)e(k)\right]$$

$$TZ\left[x_r(k-i)e(k)\right]=\frac{b}{2}\mathrm{e}^{\mathrm{j}\Phi_i}e(z\mathrm{e}^{-\mathrm{j}\omega_0})+\frac{b}{2}\mathrm{e}^{-\mathrm{j}\Phi_i}\mathrm{e}(z\mathrm{e}^{\mathrm{j}\omega_0})$$

即

$$z\hat{a}_i(z)=\hat{a}_i(z)+\frac{\delta b}{2}[\mathrm{e}^{\mathrm{j}\Phi_i}\mathrm{e}(z\mathrm{e}^{-\mathrm{j}\omega_0})+\mathrm{e}^{-\mathrm{j}\Phi_i}\mathrm{e}(z\mathrm{e}^{\mathrm{j}\omega_0})]$$

$$\hat{a}_i(z)=\frac{\delta b}{2(z-1)}[\mathrm{e}^{\mathrm{j}\Phi_i}\mathrm{e}(z\mathrm{e}^{-\mathrm{j}\omega_0})+\mathrm{e}^{-\mathrm{j}\Phi_i}e(z\mathrm{e}^{\mathrm{j}\omega_0})]$$

则滤波器输出 $Y(z)$ 的表达式如下：

$$Y(z)=TZ\left[\frac{b}{2}\sum_{i=0}^{n-1}\hat{a}_i(k)\left(\mathrm{e}^{\mathrm{j}(\omega_0 k+\Phi_i)}+\mathrm{e}^{-\mathrm{j}(\omega_0 k+\Phi_i)}\right)\right]$$

$$Y(z)=\frac{b}{2}\sum_{i=0}^{n-1}\left[\mathrm{e}^{\mathrm{j}\Phi_i}\hat{a}_i\mathrm{e}(z\mathrm{e}^{-\mathrm{j}\omega_0})+\mathrm{e}^{-\mathrm{j}\Phi_i}\hat{a}_i\mathrm{e}(z\mathrm{e}^{\mathrm{j}\omega_0})\right]$$

将前面的估计系数表达式代入，得：

$$Y(z)=\frac{b}{2}\sum_{i=0}^{n-1}\left\{e^{j\Phi_i}\frac{\delta b}{2(ze^{-j\omega_0}-1)}\left[e^{j\Phi_i}e(ze^{-2j\omega_0})+e^{-j\Phi_i}e(z)\right]+e^{-j\Phi_i}\frac{\delta b}{2(ze^{j\omega_0}-1)}\left[e^{-j\Phi_i}e(ze^{2j\omega_0})+e^{j\Phi_i}e(z)\right]\right\}$$

即

$$Y(z)=\frac{\delta nb^2}{4}\left[\frac{1}{ze^{-j\omega_0}-1}+\frac{1}{ze^{j\omega_0}-1}\right]e(z)+\frac{\delta b^2}{4}\sum_{i=0}^{n-1}\left[\frac{e^{2j\Phi_i}e(ze^{-2j\omega_0})}{ze^{-j\omega_0}-1}+\frac{e^{-2j\Phi_i}e(ze^{2j\omega_0})}{ze^{j\omega_0}-1}\right]$$

这个式子可以分为 2 项：

- 第 1 项

$$Y(z)=\frac{\delta nb^2}{4}\left(\frac{1}{ze^{-j\omega_0}-1}+\frac{1}{ze^{j\omega_0}-1}\right)e(z)$$

- 第 2 项

$$\frac{\delta b^2}{4}\sum_{i=0}^{n-1}\left(\frac{e^{2j\Phi_i}e(ze^{-2j\omega_0})}{ze^{-j\omega_0}-1}+\frac{e^{-2j\Phi_i}e(ze^{2j\omega_0})}{ze^{j\omega_0}-1}\right)$$

如果考虑到

$$\sum_{i=0}^{n-1}e^{2j\Phi_i}=\sum_{i=0}^{n-1}e^{2j\Phi-2j\omega_0 i}=e^{2j\Phi}\sum_{i=0}^{n-1}e^{-2j\omega_0 i}$$

其中

$$\sum_{i=0}^{n-1}e^{-2j\omega_0 i}=\frac{1-e^{-2jn\omega_0}}{1+e^{-2j\omega_0}}=e^{-j(n-1)\omega_0}\frac{\sin(n\omega_0)}{\sin(\omega_0)}$$

同理，得

$$\sum_{i=0}^{n-1}e^{2j\omega_0 i}=\frac{1-e^{2jn\omega_0}}{1+e^{2j\omega_0}}=e^{j(n-1)\omega_0}\frac{\sin(n\omega_0)}{\sin(\omega_0)}$$

则第 2 项可以写成

$$\frac{\delta b^2}{4}\frac{\sin(n\omega_0)}{\sin(\omega_0)}\left(\frac{e^{2j\Phi_i}e(ze^{-2j\omega_0})}{ze^{-j\omega_0}-1}+\frac{e^{-2j\Phi_i}e(ze^{2j\omega_0})}{ze^{j\omega_0}-1}\right)e(z)$$

如果 n 的取值满足下式，则第 2 项可以忽略：

$$\frac{\sin(n\omega_0)}{\sin(\omega_0)}=0$$

则 $T(z)$ 为：

$$T(z)=\frac{Y(z)}{E(z)}=\frac{\delta nb^2}{2}\left(\frac{z\cos(\omega_0)-1}{z^2-2z\cos(\omega_0)+1}\right)$$

闭环 $T_{BF}(z)=\dfrac{E(z)}{X(z)}$：

$$T_{BF}(z)=\frac{1}{1+T(z)}=\frac{z^2-2z\cos(\omega_0)+1}{z^2+\left(\frac{\delta nb^2}{2}-2\right)z\cos(\omega_0)+1-\frac{\delta nb^2}{2}}$$

由此得到一个中心频率等于 $\dfrac{\omega_0}{2\pi}$ 的抑制滤波器的传递函数。

文件 Canel_interfer_lms.m 可以验证这个例子，其中 n 阶滤波器的仿真取了 N 个点。一

个方差为 0.5 的随机信号作为输入信息 $s(k)$。正弦干扰信号如下：

$$b(k)=0.5\sin\left(\frac{2\pi fk}{N}+\frac{\pi}{3}\right)$$

则参考信号为：

$$x_r(k)=\sin\left(\frac{2\pi fk}{N}+\frac{\pi}{4}\right)$$

Canel_interfer_lms.m file

```
% Interference cancellation

N = 2000;              % Iterations number
n = 100;               % Adaptive filter order
k = 20;                % Number of interference periods
delta = 0.005;         % Adaptation coefficient

% Signals generation
for i = 1:N,
  s = 0.5*randn(1,N);
  b(i) = 0.5*sin(pi/3+k*2*pi*i/N);
  xr(i) = sin(pi/4+k*2*pi*i/N);
  x(i) = s(i)+b(i);
end

h = zeros(N,n);       % Estimated coefficients matrix
e = zeros(1,n);       % Estimation errors matrix
y = zeros(1,n);       % Filter output

% N iterations filter calculation loop
for i = n:N-1
  for j = 1:n
    y(1,i) = h(i,j)*xr(i-j+1);
    h(i+1,j) = h(i,j)+delta*xr(i-j+1)*e(1,i-1);
  end
  e(1,i) = x(i)-y(1,i);
end

% Results graphs
figure(1)
plot(1:N,x(1:N)), grid
title('x(k) Input signal in V')
xlabel('Samples')
figure(2)
plot(1:N,xr(1:N),'r'), grid
axis([0 2000 -1.2 1.2])
title('xr(k) reference signal in V')
xlabel('Samples')
figure(3)
plot(1:N-1,y(1:N-1)), grid
title('y(k) filter output signal in V')
xlabel('Samples')
```

```
figure(4)
plot(1:N-1,e(1:N-1),'r'), grid
title('Signal cleared from the e(k) interference in V')
xlabel('Samples')

figure(5)
plot(1:N,h(1:N,1)), hold on
plot(1:N,h(1:N,50)), hold off
grid
title(' h(1) and h(50) coefficients evolution')
xlabel('Samples')
% Transfer function
num = fliplr(h(N,:));
figure(6), sys = tf(num,1,1e-3);
sys1 = tf(1,1,1e-3); sys2 = feedback(sys1,sys);
bode(sys2), xlabel('Frequency in rad/s'),
ylabel('Phase in degrees ; Module in dB')
title('Synthesized transfer function')
```

在输入随机信号中可以辨别出正弦干扰信号（见图 5-9）。

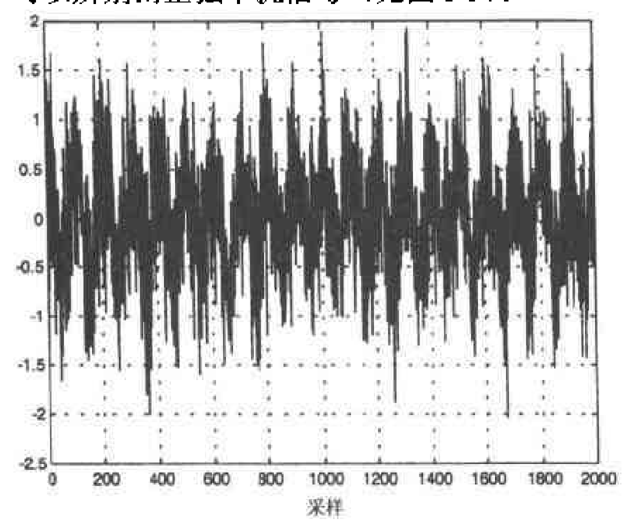


图 5-9　系统输入信号 $x(t)$

参考信号和干扰信号同频率，但振幅和相位不同（见图 5-10）。

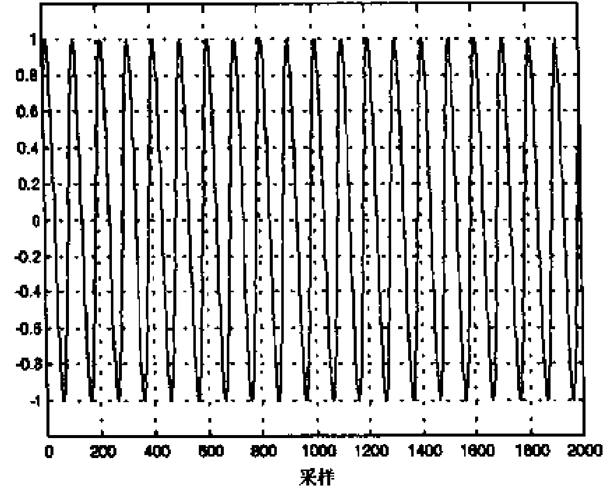


图 5-10　系统参考信号 $x_r(t)$

滤波器的输出调整自身以指数包络线的形式与干扰信号一致。由于自适应系数 δ =0.005 很小，稳态设置必需的振荡次数很重要（见图 5-11）。

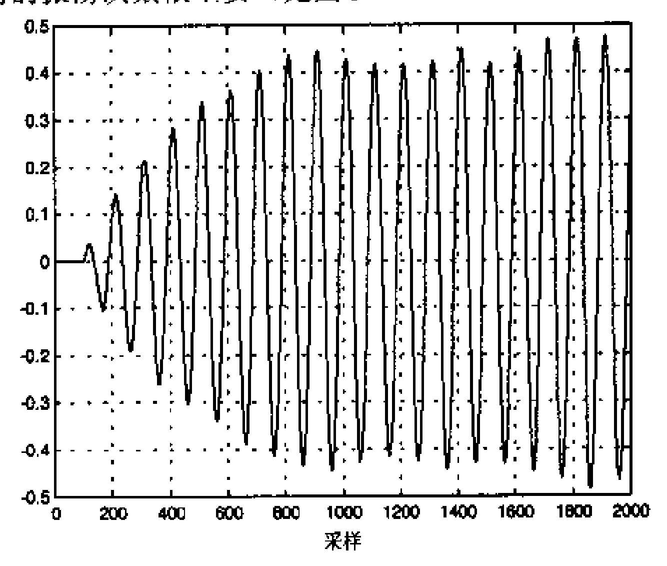


图 5-11 滤波器输出信号 $y(k)$

干扰信号持续的时间和输出信号一样（见图 5-12）。

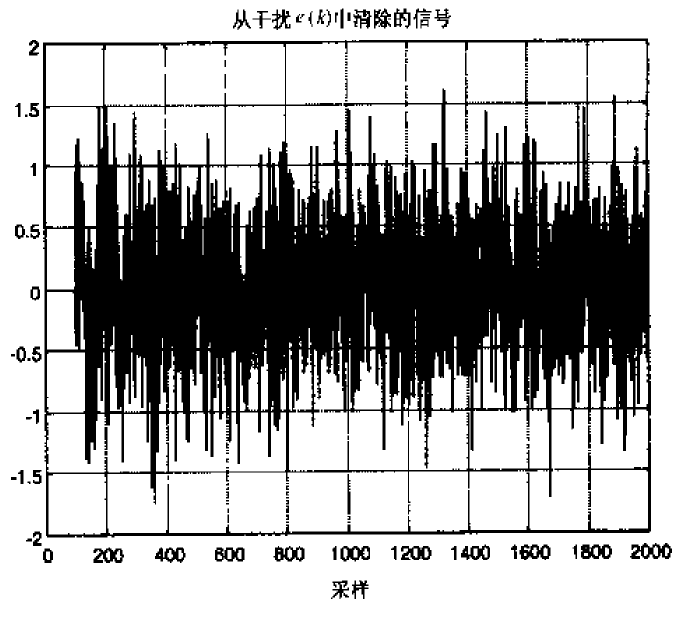


图 5-12 干扰信号

图 5-13 中的两个滤波器系数曲线显示在第 1200 个采样时刻才收敛。

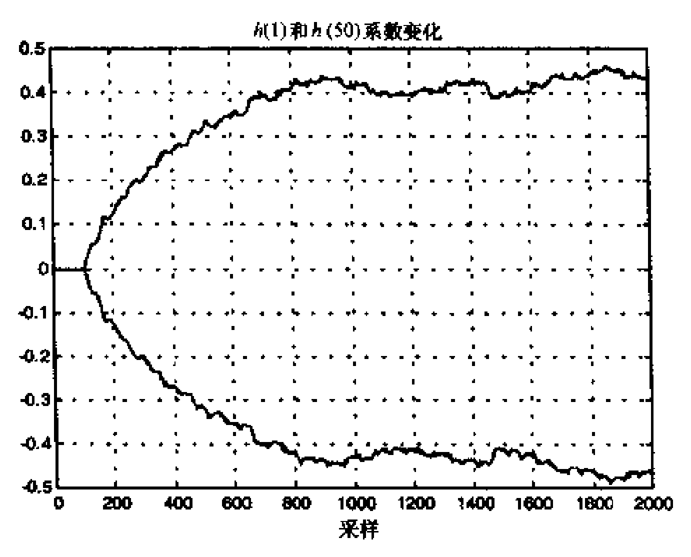


图 5-13　滤波器系数曲线

所以滤波器综合传递函数具有抑制型响应。干扰周期 k=20 时，振荡次数 N=2 000，即每周期振荡 100 次。系统离散的采样周期 T_s=1 ms，即干扰频率为 10 Hz，或角频率 ω_0=62.8 rad/s。事实上，由幅频特性图 5-14 可以看出，最大的衰减 30 dB 在 60 rad/s 和 70 rad/s 之间。

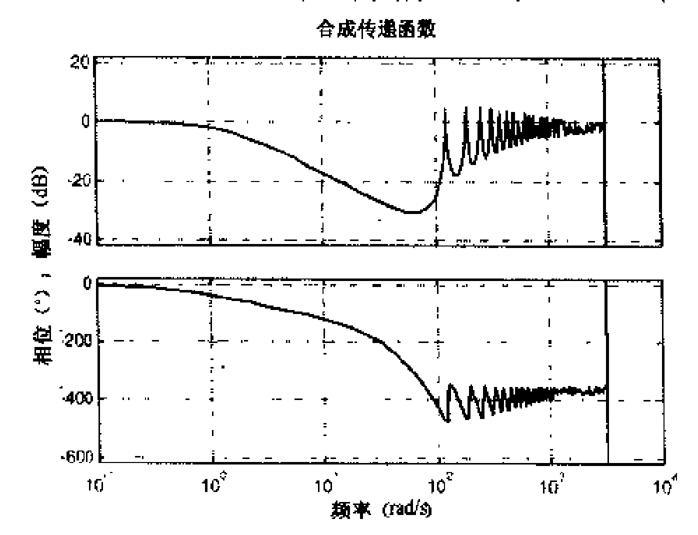


图 5-14　滤波器传递函数的幅频特性和相频特性

5.4.3　从噪声中提取信号

与上一节的例子相反，在本例中，输入信息是一个被随机信号严重干扰的正弦信号。信息是由输入信号的振幅传递的，以同频率的正弦信号作为理想参考信号（见图 5-15）。

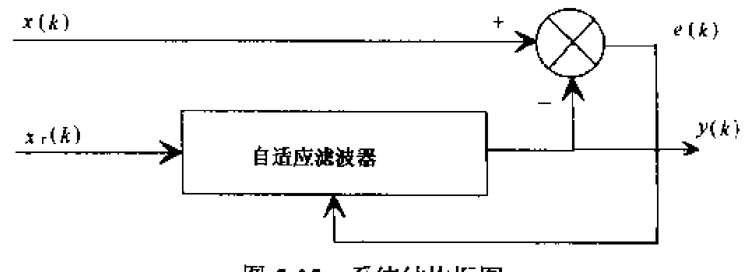


图 5-15　系统结构框图

$$x(k) = A_0 \cos(\omega_0 t + \Phi_0) + b(k)$$

其中 $b(k)$是附加的白噪声。

$$x_r(k) = A\cos(\omega_0 t + \Phi_0)$$

由于噪声和 $x_r(k)$不相关，输出 $y(k)$必须趋向于 $A_0 \cos(\omega_0 t + \Phi_0)$ 才可以消除误差。这时，自适应滤波器必须取中心频率为 $\frac{\omega_0}{2\pi}$ 的通频带型传递函数。

文件 Exract_sinus_lms.m 为本例的仿真，其中滤波器阶数 n=200，采样周期为 1ms。

Exract_sinus_lms.m file

```
% Extraction of a sinusoidal signal drowned in a white noise
N = 1000;             % Sequence length
n = 200;              % Filter order
k = 12;               % Number of sinus periods
delta = 0.001;        % Adaptation factor
Te = 1e-3 ;           % Sample period

% Signals generation

b = 0.8*randn(1,N);          % White noise generation
for i = 1:N
   xr(1,i) = sin(k*2*pi*i/N);    % Reference signal
   x(1,i) = xr(1,i)+ b(i);   % Filter input
end

% N steps filter calculation loop
h = zeros(N+1,n);
e(n) = 0;

for i = n+1:N
   y(i) = 0;
   for j = 1:n
      y(i) = y(i)+h(i,j)*x(1,i-j+1);
      h(i+1,j) = h(i,j)+delta*e(i-1)*x(1,i-j+1);
   end
   e(i) = xr(1,i)-y(i);
end

% Results graphs

figure(1)
plot(0:N-n,x(1,n:N)), grid
title('x(k) input signal in V')
xlabel('Samples')

figure(2)
plot(0:N-n,xr(1,n:N),'r') , grid
axis([0 800 -1.2 1.2])
title('xr(k) reference signal in V')
xlabel('Samples')
```

```
figure(3)
plot(0:N-n,e(n:N)), hold on
plot(0:N-n,y(n:N),'r'), hold off
grid
title('e(k) error and y(k) output in V')
xlabel('Samples')
gtext('e(k)'), gtext('y(k)')

figure(4)
plot(0:N-n,h(n:N,1)), hold on
plot(0:N-n,h(n:N,2),'r'), hold off
grid
title('a(n-1) and a(n-2) coefficients evolution')
xlabel('Samples')

% Filter transfer function

figure(5)
num1 = fliplr(h(N,:));
sys1 = tf(num1,1,Te);

bode(sys1), hold off
xlabel('Frequency in rad/s')
ylabel('Phase in degrees ; Modulus in dB')
title('Synthesized filter')
```

从方差等于 0.793 的白噪声中提取出了正弦信号。

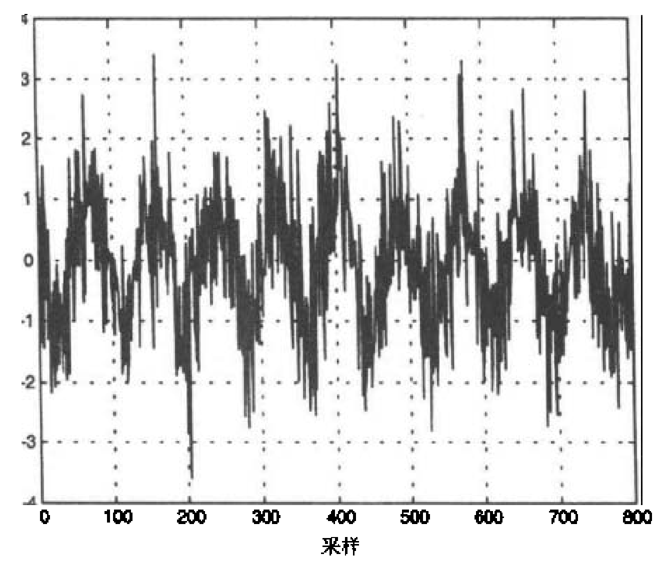


图 5-16 输入信号 $x(k)$

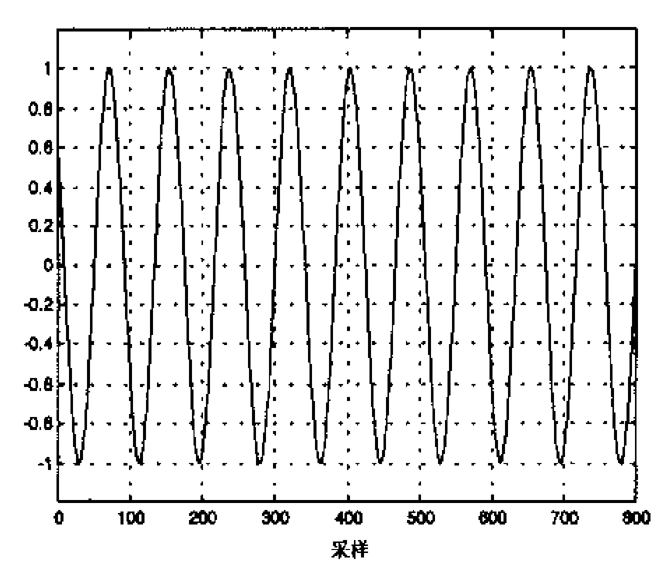


图 5-17　参考信号 $x_r(k)$

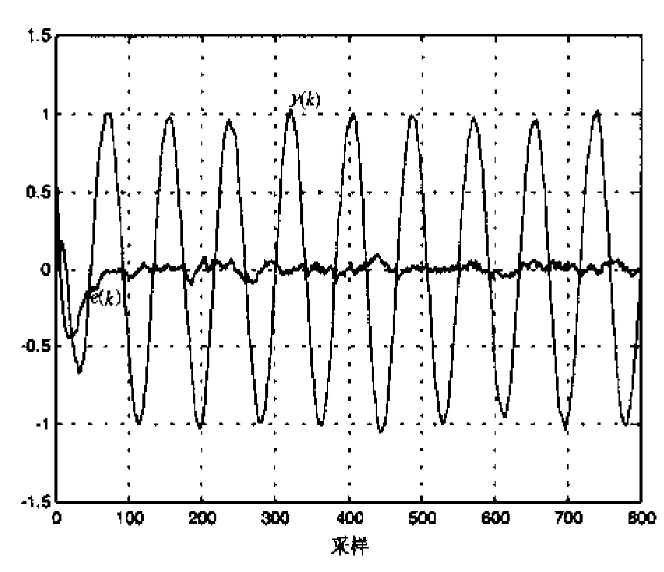


图 5-18　误差信号 $e(k)$和输出信号 $y(k)$

经过约 200 次振荡以后得到了没有噪声的输出信号，相应于滤波器阶数 n=200 的理论最短持续时间。

取最近 500 个采样点，用指令 std 计算得残差的标准偏差等于 0.028，噪声功率减少到了 1/29。

图 5-19 为滤波器系数的变化。

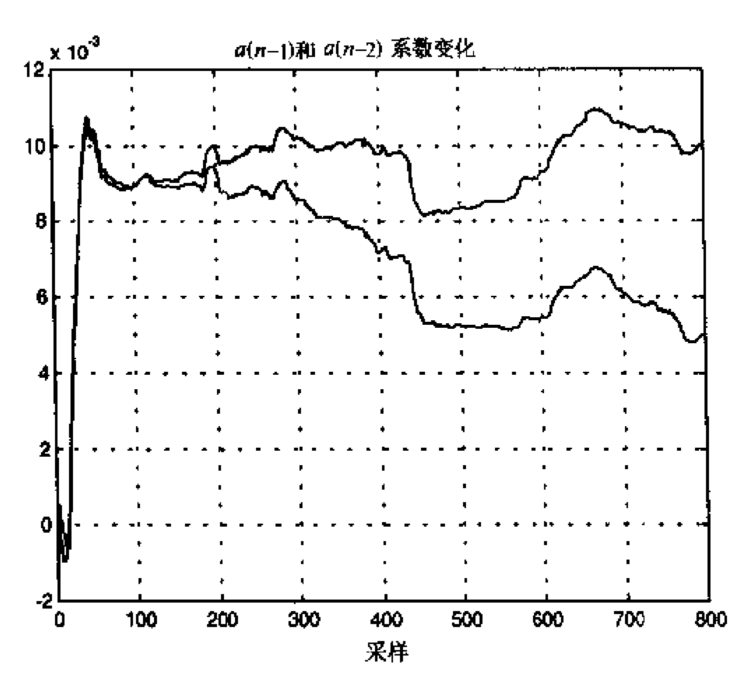


图 5-19　滤波器系数变化曲线

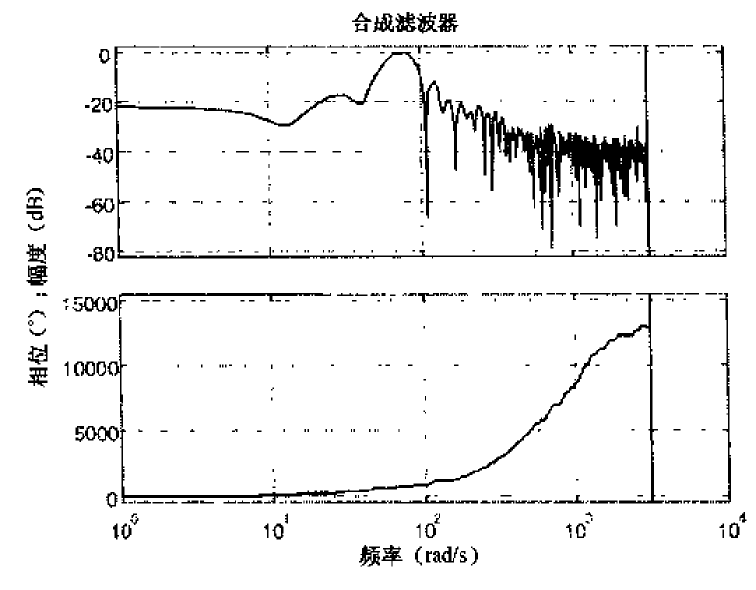


图 5-20　滤波器传递函数的幅频特性和相频特性

取正弦信号的 12 个周期，1000 个点，即每个周期 83.3 个点。采样周期是 1 ms，它产生一个 12 Hz 的正弦信号频率。可以看到，滤波器传递函数具有 70~80 rad/s 的通频特性，相应于期望函数。现在修正正弦信息信号，使它的频率在采样时刻 $N/2=500$ 以后从 75 rads/s 变到 150 rads/s，这主要用来检验滤波器相对于一个不固定信息信号的自适应性。

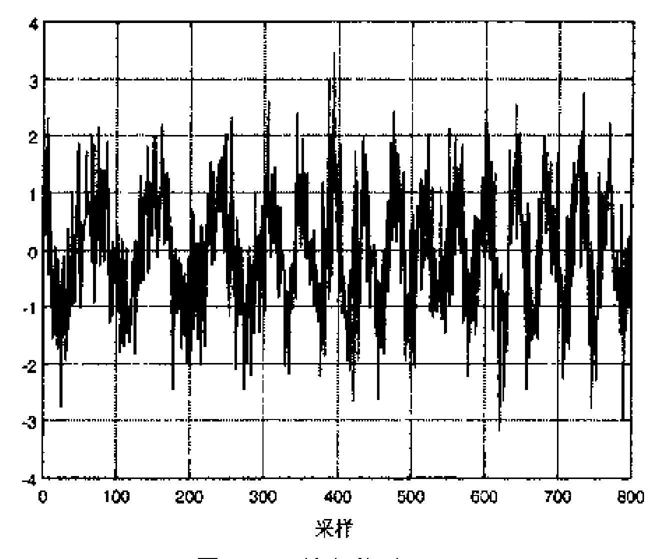


图 5-21　输入信号 $x(k)$

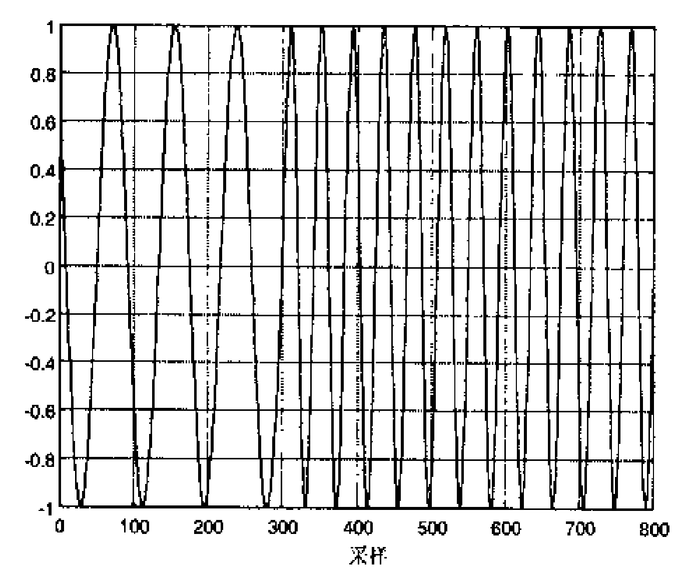


图 5-22　参考信号 $x_1(k)$

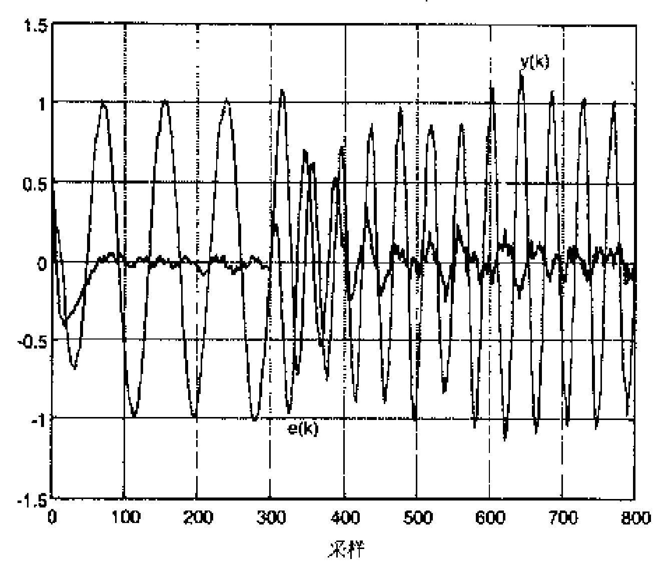


图 5-23　误差信号 $e(k)$和滤波器输出信号 $y(k)$

滤波器在大约 200 步以后适应了输入信息的频率变化，并且因系数变化而产生了一个具有双中心频率的通频带滤波器（见图 5-24）。

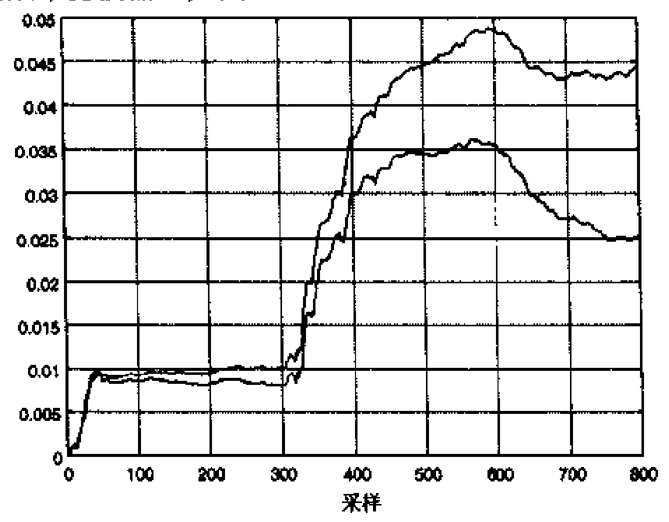


图 5-24 滤波器系数 $a(n\text{-}1)$和 $a(n\text{-}2)$变化曲线

第 1 个传递函数表示 $N/2-1$ 步即频率变化以前的滤波器状态。第 2 个传递函数表示 N 步时得到的滤波器状态，我们看到中心频率已由 75 rads/s 变到了 150 rads/s。由于具有固定的自适应增益，LMS 算法适应于不固定的输入信号。但这个优点是以一个不可忽略的残差为代价的。

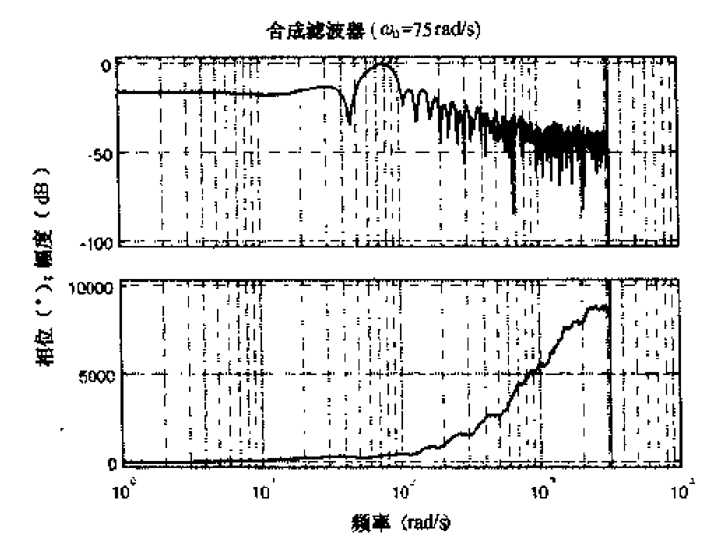


图 5-25 滤波器传递函数的幅频特性和相频特性

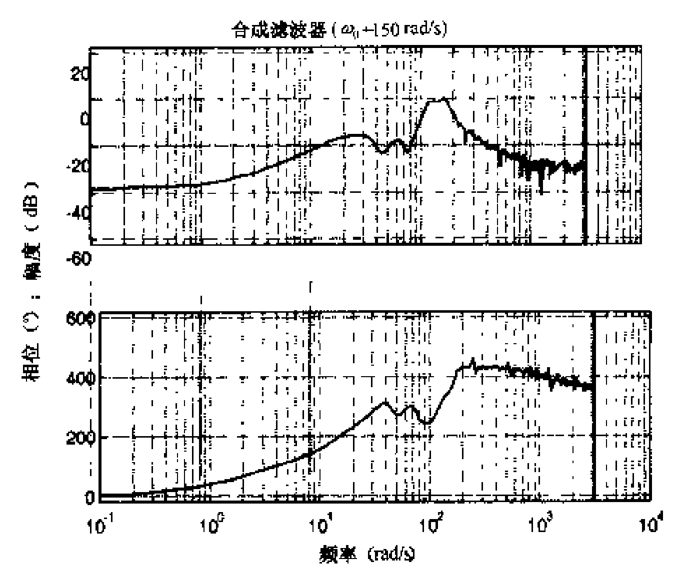


图 5-26　滤波器传递函数的幅频特性和相频特性

5.5　RLS 自适应滤波器举例

5.5.1　从噪声中提取信号

这里仍以 5.4.3 节中研究过的 LMS 滤波器为例，在相同条件下研究，即具有相同的步数、滤波器阶数、正弦信号周期数。仍以下面的信号作为自适应滤波器的输入信号：

$$x(k)=A_0\cos(\omega_0 t+\Phi_0)+b(k)$$

其中 $b(k)$是附加的白噪声。

$$x_r(k)=A\cos(\omega_0 t+\Phi_0)$$

应用于 RLS 自适应滤波器的算法可用下列方程描述：

$$g(k)=\frac{C_{xx}^{-1}(k-1)x(k)}{1+x^{\mathrm{T}}(k)C_{xx}^{-1}(k-1)x(k)}$$

$$e(k)=y(k)-h^{\mathrm{T}}(k-1)x(k)$$

$$h(k)=h(k-1)+g(k)e(k)$$

$$C_{xx}^{-1}(k)=C_{xx}^{-1}(k-1)-g(k)x^{\mathrm{T}}(k)C_{xx}^{-1}(k-1)$$

$$y(k)=h^{\mathrm{T}}(k)x(k)$$

其中：

- $g(k)$　自适应增益行向量，大小（1,n）；
- $e(k)$　先验误差；
- $h(k)$　自适应滤波器系数行向量，大小(1,n)；
- $C_{xx}(k)$　输入信号 $x(k)$的自相关转置矩阵，大小（1,n）；
- $y(k)$　自适应滤波器输出；
- $x(k)$　行向量，自适应滤波器输入，大小（1,n）。

自相关逆矩阵初始化为 10000I，I 为单位矩阵（n, n）。系数向量的初始值等于 0。所

研究的滤波器阶数为 200，采样周期等于 1 ms。

Extract_sinus_rls.m file

```
% Extraction of a sinusoidal signal drowned in a white noise

N = 1000;            % Sequence length
n = 200;             % Filter order
k = 12;              % Number of sinus periods
Ts = 1e-3;           % Sample period

% Signals generation

b = 0.8*randn(1,N);                        % White noise generation
for i = 1:N
     xr(1,i) = sin(k*2*pi*i/N);            % Reference signal
     x(1,i) = xr(1,i)+ b(i);               % Filter input
end

% Initializations

Cxx = 10000*eye(n);
g = zeros(N,n);
h = zeros(N,n);
e = zeros(1,N);
y = zeros(1,N);
tr = zeros(1,N);

% N steps filter calculation loop

for i = n+1:N,
  g(i,:) = (Cxx*x(i-n+1:i)'/(1+x(i-n+1:i)*Cxx*x(i-n+1:i)'))';
  % Adaptation gain
  e(1,i) = xr(i)-h(i-1,:)*x(i-n+1:i)';  % A priori error
  h(i,:) = h(i-1,:)+e(1,i)*g(i,:);      % Coefficients
  Cxx = Cxx-g(i,:)'*x(i-n+1:i)*Cxx;     % Correlation matrix
  y(1,i) = h(i,:)*x(i-n+1:i)';          % Filter output
  tr(1,i) = trace(Cxx);
end

% Results graphs

figure(1)
plot(0:N-n,x(1,n:N)), grid
title('x(k) input signal in V')
xlabel('Samples')

figure(2)
plot(0:N-n,xr(1,n:N),'r')
grid
axis([0 800 -1.2 1.2])
title('xr(k) reference signal in V')
```

```
xlabel('Samples')

figure(3)
plot(0:N-n,e(1,n:N)), hold on
plot(0:N-n,y(1,n:N),'r'), hold off
grid
title('e(k) error and y(k) output in V')
xlabel('Samples')
gtext('e(k)'), gtext('y(k)')

figure(4)
plot(0:N-n,h(n:N,1)), hold on
plot(0:N-n,h(n:N,2),'r'), hold off
grid
title('a(n-1) and a(n-2) coefficients evolution')
xlabel('Samples')

% Filter transfer function

figure(5)
num1 = fliplr(h(N,:));
sys1 = tf(num1,1,Ts);
bode(sys1), hold off
title('Synthesized filter')
xlabel('Frequency in rad/s')
ylabel('Phase in degrees ; Module in dB')

figure(6)
semilogy(0:N-n,tr(n:N)), grid
title('Cxx matrix trace')
xlabel('Samples')
```

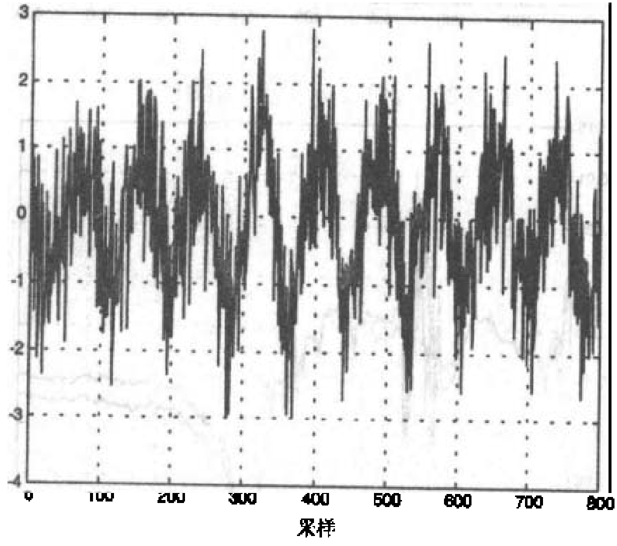


图 5-27　输入信号 $x(k)$

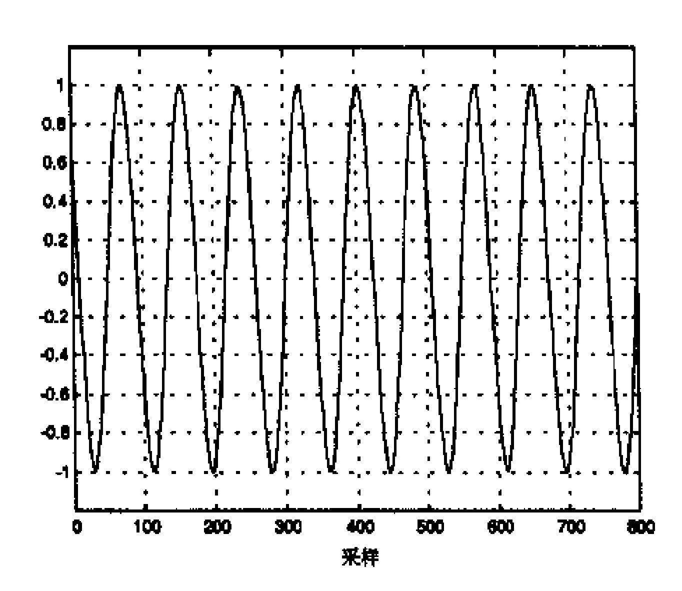


图 5-28　参考信号 $x_r(k)$

可以看到，误差 $e(k)$曲线与 LMS 滤波器类似。在约 30 步以后误差 $e(k)$收敛。

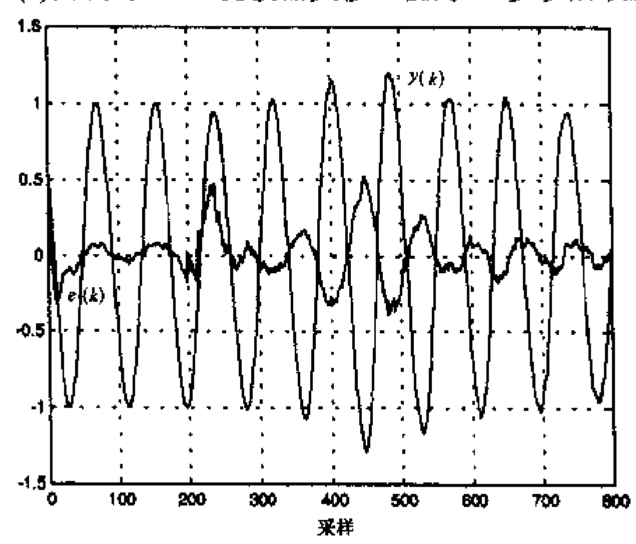


图 5-29　误差 $e(k)$和输出 $y(k)$信号

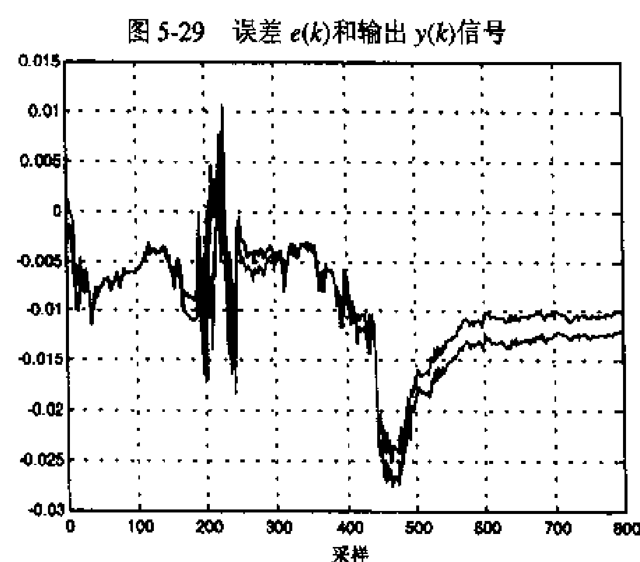


图 5-30　滤波器系数 $a(n-1)$和 $a(n-2)$变化曲线

系数的变化曲线在约 200 步时有一个超调，这是由于 $h(k)$向量为零，所以 200 步以后仅代表 x 值。获得的滤波器的传递函数也类似于 LMS 滤波器的传递函数，相应的预测也类似。它的中心频率调整为正弦信号频率，即 75 rad/s（见图 5-31）。

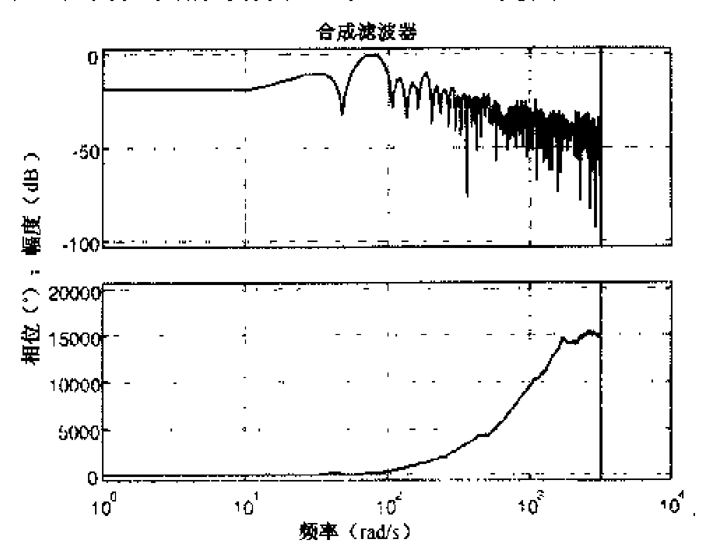


图 5-31　合成滤波器传递函数的幅频特性和相频特性

C_{xx} 曲线在采样步数 n=200 时突变。200 步以后，曲线值变小，使滤波器不能够再根据输入信号的统计变换进行调整（见图 5-32）。

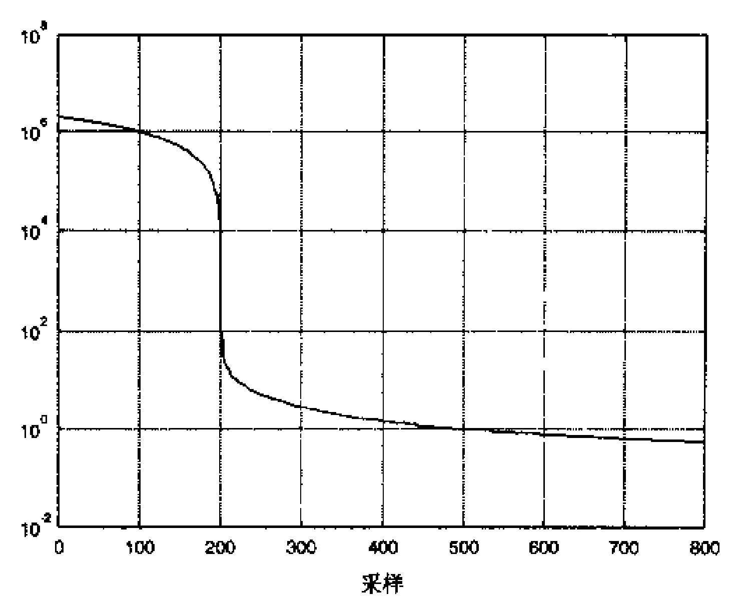


图 5-32　C_{xx}矩阵曲线

这个现象可以通过在 500 步以后，使输入信号频率增加 1 倍来验证。误差 $e(t)$和输出 $y(t)$的曲线清楚地表明滤波器很难调整自己以适应 $x(t)$的频率变化。

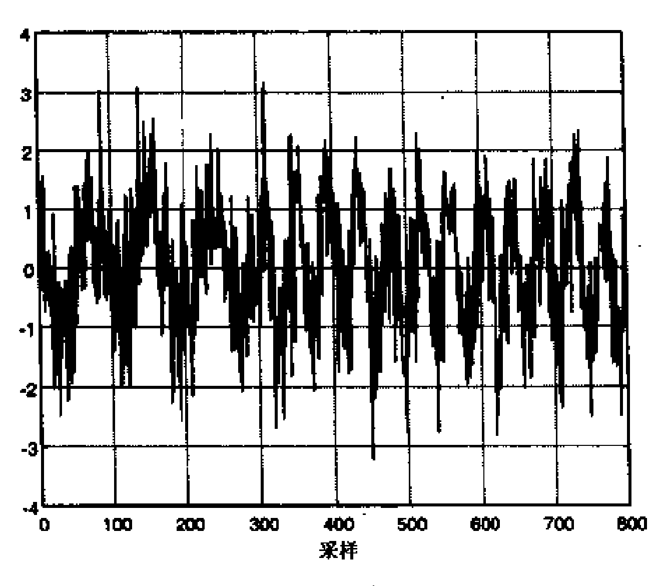


图 5-33　输入信号 $x(k)$

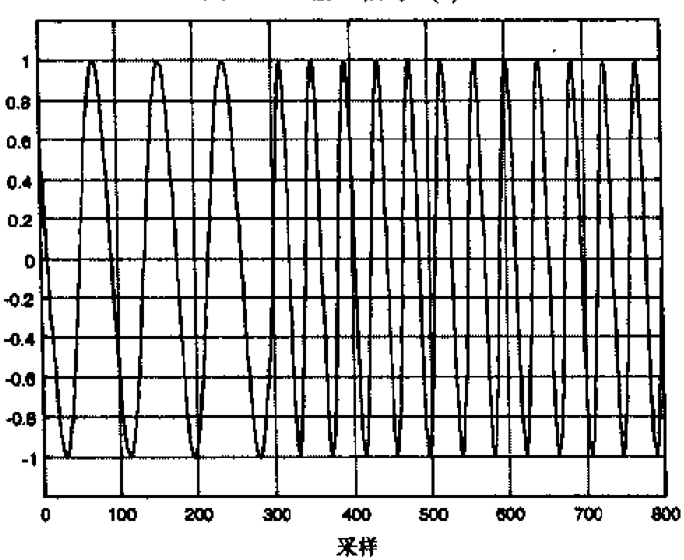


图 5-34　参考信号 $x_r(k)$

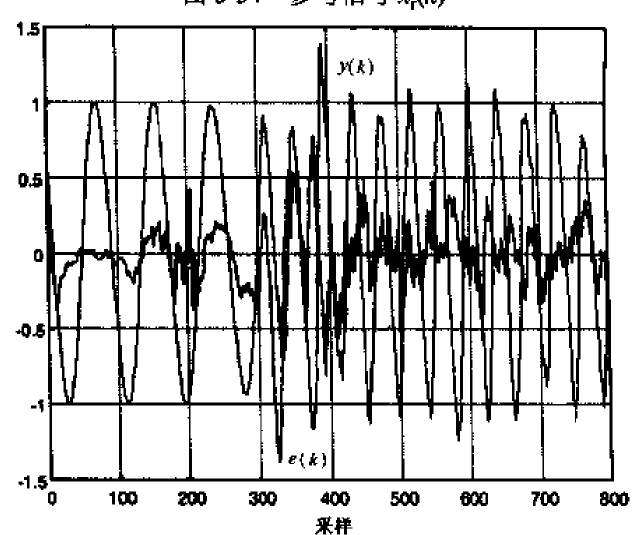


图 5-35　误差 $e(k)$和输出 $y(k)$信号

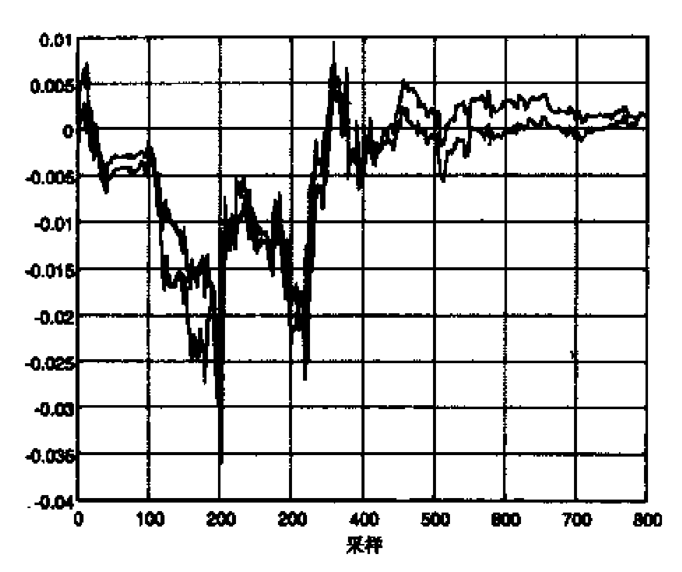


图 5-36　滤波器系数 $a(n-1)$和 $a(n-2)$变化曲线

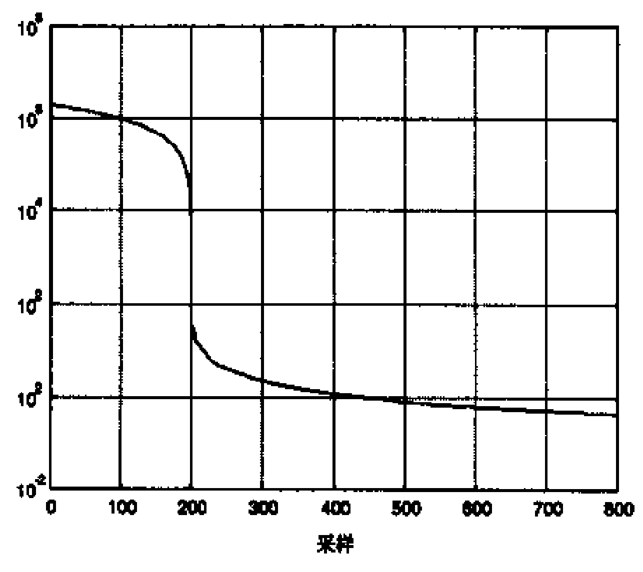


图 5-37　C_{xx} 曲线

在频率改变时，C_{xx}矩阵的值小于 1，不足以形成对输入信号变化的自适应校正。

下面的文件 Extract_sinus_rlsl.m 仿真了一个具有固定轨迹的 40 阶 RLS 滤波器，C_{xx}矩阵总初始化为 10000I，但在采样的每一步，它总是大于或等于 100。如果不是这样，就调整到 100。输入和参考信号与前面的仿真一样。因此我们得到一个初始化的值能够保证快速收敛，并有固定轨迹，确保具有自适应校正性能的滤波器。

Extract_sinus_rlsl.m file

```
% Extraction of a sinusoidal signal drowned in a white noise
% RLS filter, constant trace
N = 1000;            % Sequence length
n = 40;              % Filter order
k = 12;              % Sinus periods number
Ts = 1e-3;           % Sample period

% Signals generation
```

```
b = 0.8*randn(1,N);            % White noise generation
for i = 1:N
  if i<N/2
    xr(1,i) = sin(k*2*pi*i/N);% Reference signal
  else
    xr(1,i) = sin(k*4*pi*i/N);
  end
  x(1,i) = xr(1,i)+ b(i);      % Filter input
end

% Initializations
Cxx = 10000*eye(n);
g = zeros(N,n); h = zeros(N,n); e = zeros(1,N);
y = zeros(1,N); tr = zeros(1,N);

% N steps filter calculation loop
for i = n+1:N
  g(i,:)= (Cxx*x(i-n+1:i)'/(1+x(i-n+1:i)*Cxx*x(i-n+1:i)'))';
  % Adaptation gain
  e(1,i) = xr(i)-h(i-1,:)*x(i-n+1:i)';        % A priori error
  h(i,:) = h(i-1,:)+e(1,i)*g(i,:);            % Coefficients
  Cxx = Cxx-g(i,:)'*x(i-n+1:i)*Cxx;           % Correlation matrix
  y(1,i) = h(i,:)*x(i-n+1:i)';                % Filter output
  tr(1,i) = trace(Cxx);
  if tr(1,i)<100
    Cxx = 100*eye(n)*Cxx/tr(1,i);
  end
end

% Results graphs
figure(1), plot(0:N-n,y(1,n:N),'r')
title('y(k) output in V'), xlabel('Samples')
figure(2), semilogy(0:N-n,tr(n:N)), grid
title('Cxx matrix trace'), xlabel('Samples')
```

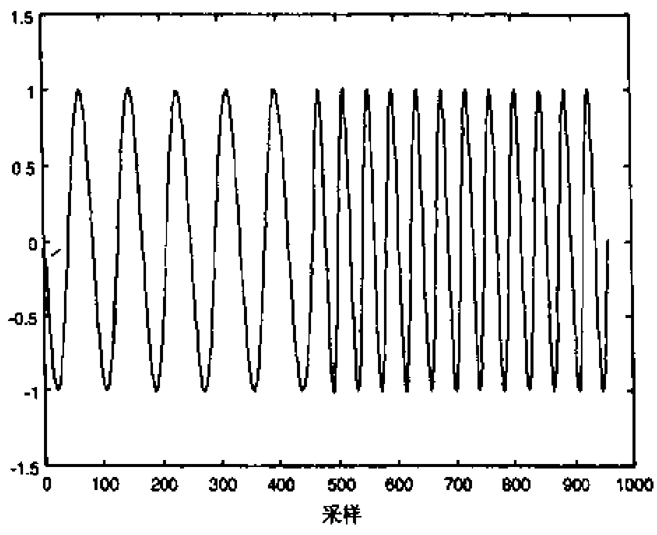


图 5-38　输出信号 $y(k)$

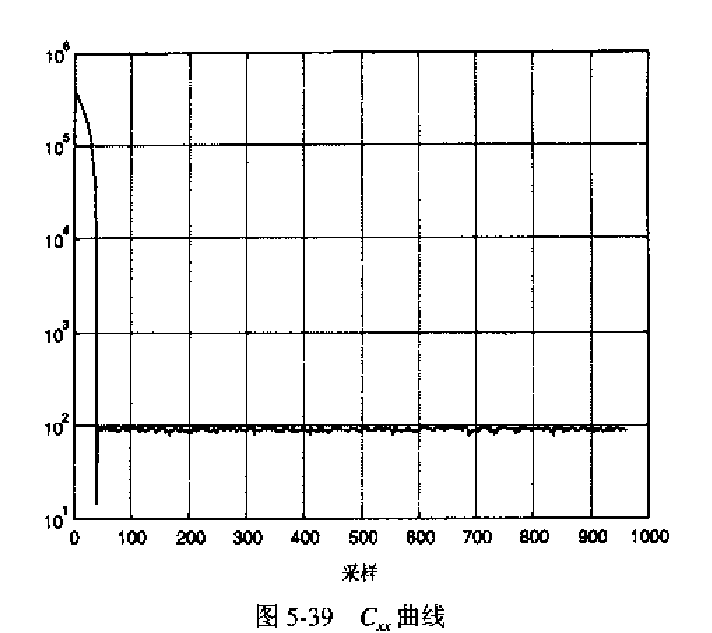


图 5-39　C_{xx} 曲线

30 步以后，轨迹就保持为常量，信号的改变得到了校正。

应用 1　功率放大器

本应用的目的是掌握使功率放大器动态工作稳定的相关技术。我们都知道，放大器和振荡器实现的区别很小。这里，我们将最大程度地把振荡器转为放大器的原理表述清楚（见图 1-1）。

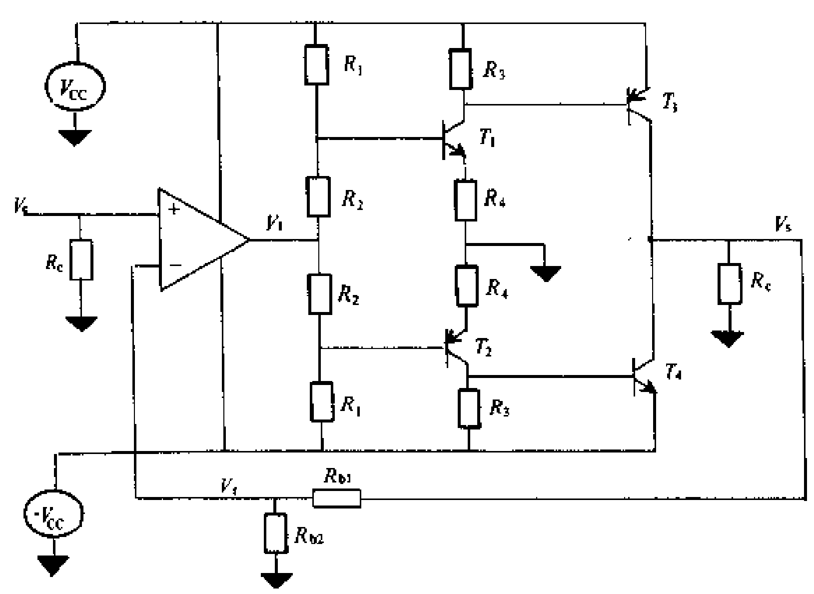


图 1-1　振荡器转为放大器原理图

1.1　放大器介绍

功率放大器可分成如下几部分：

- 获得 V_e 和 V_r 之间的电压差并将其放大的运算放大器级；
- 一个具有位于 V_r 和 V_s 之间晶体管的 B 级放大器级；
- 一个位于 V_s 和 V_r 之间的负反馈。

除了反馈带来的好处，如输入阻抗增加、输出阻抗减小、失真减小、通带增加、参数灵敏度降低），反馈使整体电压放大，更加稳定。

放大器方框图如图 1-2 所示。

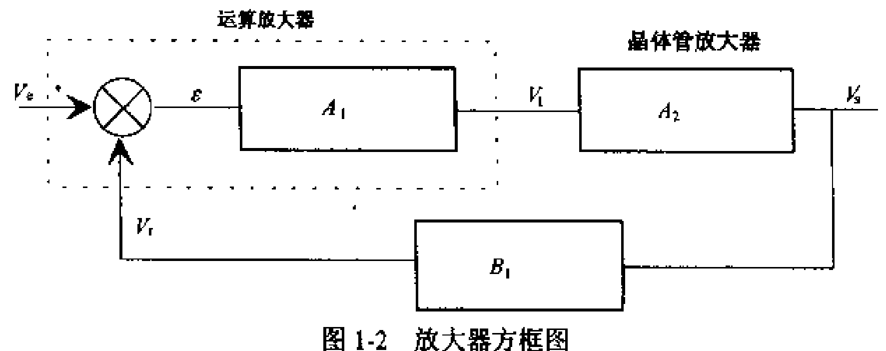


图 1-2　放大器方框图

典型的运算放大器有如下特性：

- 开环增益 A_1=1000000；
- 第 1 截止频率 f_1=10 Hz；
- 第 2 截止频率 f_2=1 MHz；
- 消耗电压 2.5 V。

晶体管放大器有如下特性：

- 开环增益 A_2=10；
- 第 1 截止频率 f_3=10 kHz；
- 第 2 截止频率很高，不予以考虑；
- 消耗电压 1.5 V。

因为放大器是 B 级，因此输出级有静态电流。这代表在±0.5 V 允许交越失真出现。开环传递函数记作

$$\mathrm{OLTF}=\frac{V_r}{\varepsilon}=A_1A_2B_1$$

闭环传递函数记作

$$\mathrm{CLTF}=\frac{V_s}{V_e}=\frac{A_1A_2}{1+A_1A_2B_1}$$

如果 $A_1A_2B_1>>1$，那么

$$\mathrm{CLTF}=\frac{1}{B_1}$$

整个放大器的 SIMULINK 模型如图 1-3 所示。

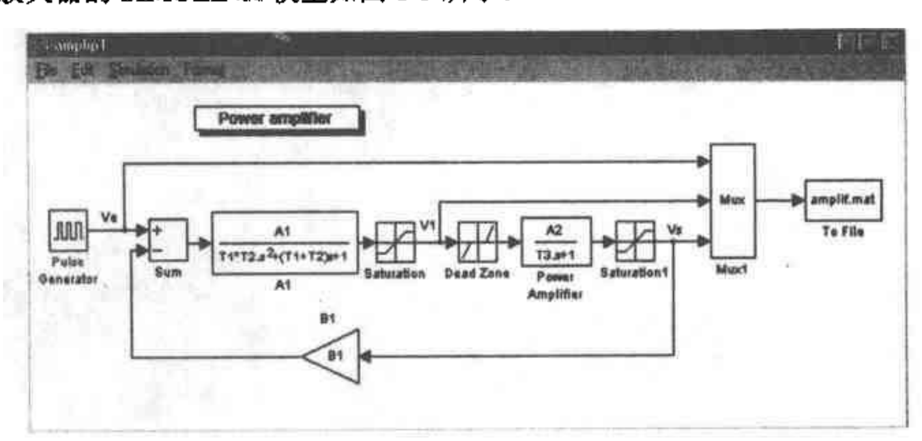


图 1-3　整个放大器的 SIMULINK 模型

图中：

$$\begin{cases} A_1=1000000 & T_1=\dfrac{1}{2\pi f_1} \quad T_2=\dfrac{1}{2\pi f_2} \\ A_2=10 & T_3=\dfrac{1}{2\pi f_3} \\ B_1=0.01 & \end{cases}$$

1.2 放大器的特性

首先检查整体的稳定性。为此，在表示开环传递函数的 Bode 图和计算系统相位与增益时要用到 bode 和 margins 功能。

文件 ampli1.m 可帮助我们找到这个信息。

ampli1.m file

```
% Power amplifier
% Open loop transfer function

% operational amplifier stage
A1=1e6;
T1=0.0159;
T2=1.59e-7;
num1=A1;
den1=[T1*T2 T1+T2 1];
sys1=tf(num1,den1);
printsys(num1,den1,'p');
```

◆ 运算放大器级的传递函数

```
num/den=
                    1000000
         ----------------------------------
         2.5281e - 009 p^2 + 0.0159 p + 1
```

◆ 晶体管级的开环传递函数

```
% Transistors stage
A2=10;
T3=1.59e-5;
num2=A2;
den2=[T3 1];
sys2=tf(num2,den2);
printsys(num2,den2,'p');

num/den=
          10
------------------
1.59e - 005 p + 1
```

◆ 整个放大器的开环传递函数

```
% Open loop transfer function
B1=0.01;
sys=sys1*sys2*B1
[z,p,k]=zpkdata(sys);
sys_k=zpk(z,p,k)

Transfer function
                    100000
----------------------------------------------
4.02e - 014 s^3 + 2.533e - 007 s^2 + v0.01592 s + 1
```

```
Zero/pole/gain:
                2.487761e + 018
----------------------------------------
(s+6.289e006) (s+6.289e004) (s+62.89)
```

这里介绍 A_1 运算放大器级、A_2 晶体管级和 OLTF 整个放大器的开环传递函数的 Bode 图。

```
% Transfer function graphs of A1, A2 and OLTF=A1*A2*B1

figure(1)
w = logspace(1 ,7,100);
bode(sys1,w), hold on
bode(sys2,w), hold on
bode(sys,w), hold off
title('Open loop transfer functions')
xlabel('Frequency in rad/s')
ylabel('Phase in degrees ; Module in dB')
gtext('\leftarrowA1'), gtext('\uparrowA2')
gtext('\uparrowOLTF')
```

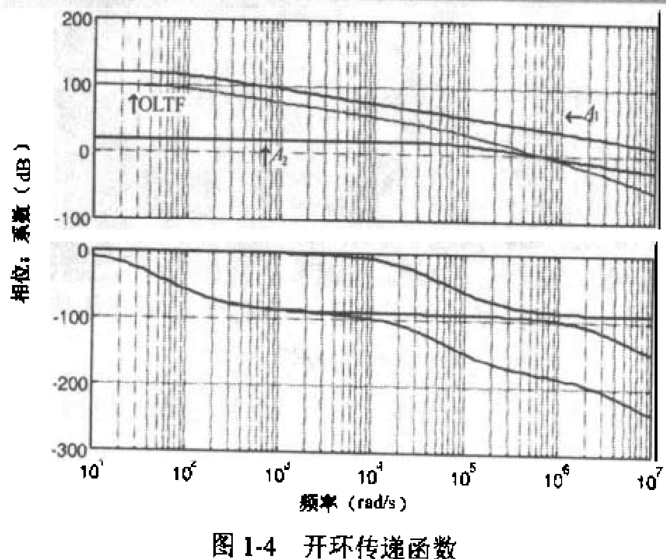


图 1-4　开环传递函数

可以看到，整个放大器的开环传递函数有 3 个截止频率，因此，总相位在大约 10Mrad/s 下趋于-270°。

相位裕度（与开环传递函数在 0dB 系数、-180°条件下比较的相位差）用大约 600krad/s 的频率测量，接近于 0，因此稳定性很重要。因为正常的话，相位裕度必须在 45°左右。增益裕度代表与开环传递函数在-180°、0 dB 条件下相比较的差别。

用 margin 函数可以得到相位、增益裕度以及相应频率。

```
%phase and gain margins of the CLTF
[mg,mp,wg,wp] = margin(sys);
disp(['Gain margin in dB = 'num2str(mg)])
disp(['Corresponding Frequency = 'num2str(wg/(2*pi))])
disp(['Phase margin in degrees = 'num2str(mp)])
disp(['Corresponding Frequency = 'num2str(wp/(2*pi))])

Gain margain in dB = 1.011
```

```
Corresponding Frequency = 100147.9849
Phase margin in degress = 0.062208
Corresponding Frequency = 99600.6595
```

在f_p=99.6 kHz 下计算的相位裕度是 0.06°。

在下面的 ampli2.m 文件的帮助下，获得模拟后的单位阶跃响应图。

ampli2.m file

```
% Power amplifier
% Closed loop transfer function
% Indicial response

load amplif.mat;
t = signal(1,:);
ve = signal(2,:);
v1 = signal(3,:);
vs = signal(4,:);

% Indicial response graph
figure(1)
plot(t,100*ve), hold on
plot(t,v1,'r:'), hold on
plot(t,vs), hold off
title('0.1 V step response')
xlabel('Time')
ylabel('Amplitude in V')
gtext('100*Ve\rightarrow'), gtext('\leftarrowV1')
gtext('\downarrowVs')
```

图 1-5 证实了系统不稳定程度。为了修正，有必要减少开环传递函数的增益。为此，考虑在二阶电阻桥帮助下的晶体管放大器级的反馈。

在图 1-5 中，可以看出，由于在晶体管级有 1.5 V 的损耗电压，输出信号 V_s 在 13.5 V 饱和。

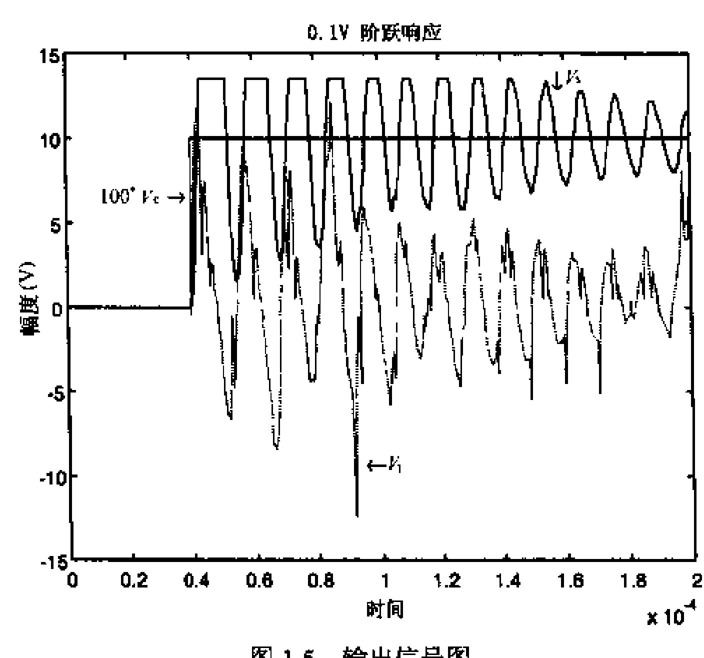


图 1-5 输出信号图

1.3 有晶体管级反馈的放大器

原理（见图 1-6）

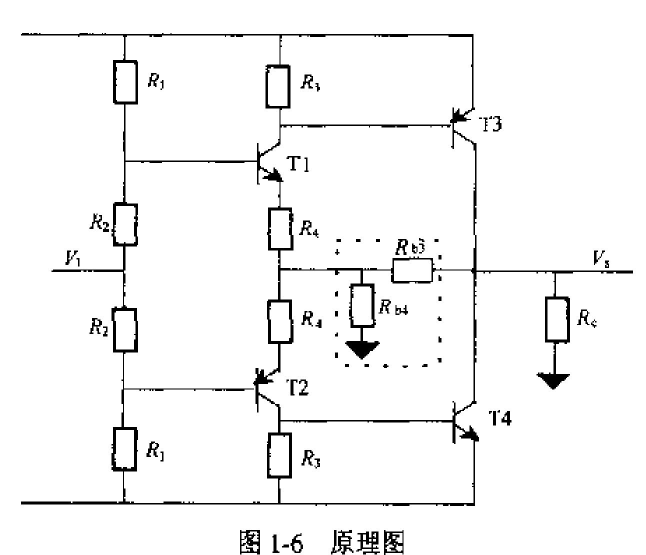


图 1-6 原理图

反馈增益 B_2 为：

$$B_2=\frac{R_{b4}}{R_{b3}+R_{b4}}$$

这样图 1-2 变为图 1-7 的形式。

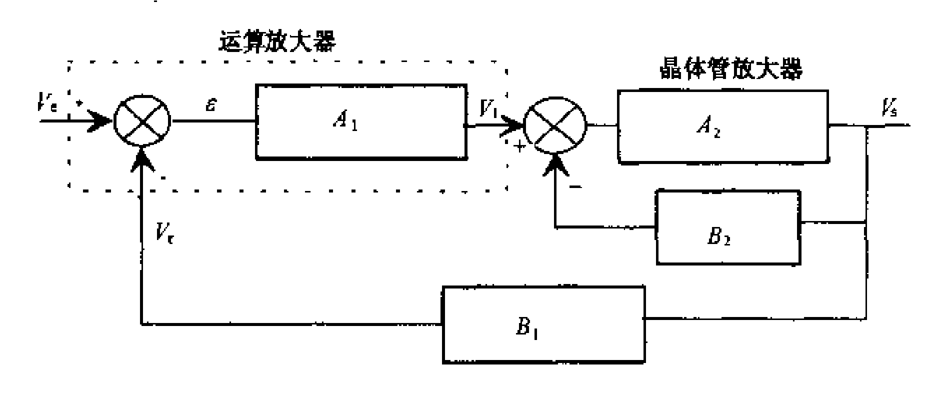


图 1-7 修改后的图

为将晶体管级放大倍数固定在 3 左右，必须降低约 10 dB 的损耗，这里让 B_2 =0.33。开环传递函数表示如下：

$$\text{OLTF}=\frac{V_r}{\varepsilon}=\frac{A_1A_2A_3}{1+A_2B_2}$$

闭环传递函数如下：

$$\text{CLTF}=\frac{V_s}{V_e}=\frac{A_1A_2}{1+A_2(B_2+A_1B_1)}$$

因此整个放大器的 SIMULINK 模型如图 1-8 所示。

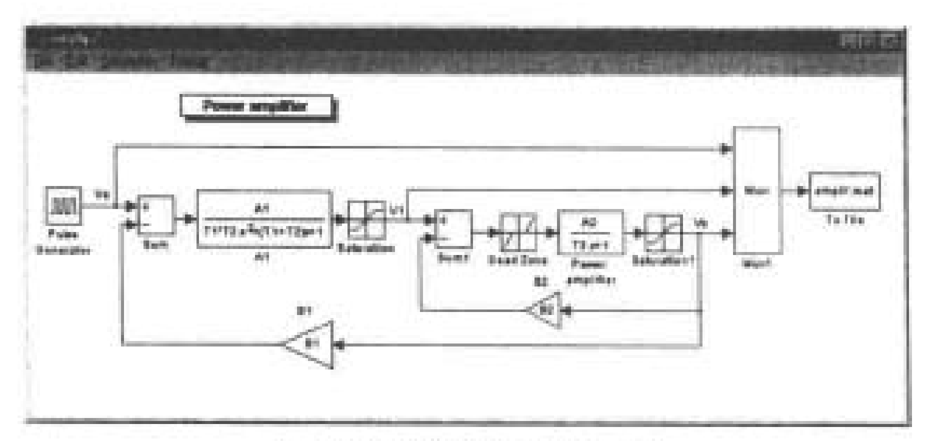


图 1-8　完整的 SIMULINK 模型图

文件 ampli3.m 能检查整个系统的稳定性。

amlip3.m file

```
% Power amplifier
% Transistors stage feedback
% Open loop transfer function
% Operational amplifier stage

A1=1e6;
T1=0.0159;
T2=1.59e-7;
num1=A1;
den1=[T1*T2 T1+T2 1];
sys1=tf(num1,den1);
printsys(num1,den1,'p');

% Transistors stage
A2=10;
T3=1.59e-5;
B2=0.33;
num2=A2;
den2=[T3 1];
sys2=tf(num2,den2);
sys2b=sys2/(1+sys2*B2);
```

◆　整个放大器的开环传递函数

```
% Open loop transfer function
B1=0.01;
sys=sys1*sys2b*B1
Transfer Function:
1.59s + 100000
-----------------------------------------------------------
6.391e - 019s^4 + 4.233e - 012s^3 + 1.351e - 006s^2 + 0.06845s + 4.3
```

◆　预示图

这里呈现 A_1 运算放大器级、A_2 晶体管级和 OLTF 整个放大器的开环传递函数的预示图。

```
% A1, A2 and OLTF=A1*A2*B1 transfer functions tracing
figure(1)
W = logspace(1 ,7,100);
```

```
bode(sys1,w), hold on
bode(sys2b,w), hold on
bode(sys,w), hold off
title('Open loop transfer functions')
xlabel('Frequency in rad/s')
ylabel('Phase in degrees ; Modulus in dB')
gtext('\leftarrowA1'), gtext('\uparrowA2'),gtext('\uparrowOLTF')
```

开环传递函数 A_2 的增益减少，A_2 和 OLTF 曲线向下移动。然而，A_2 的截止频率因为增益/通带保持不变而增加，这补偿了研究结果的一部分。

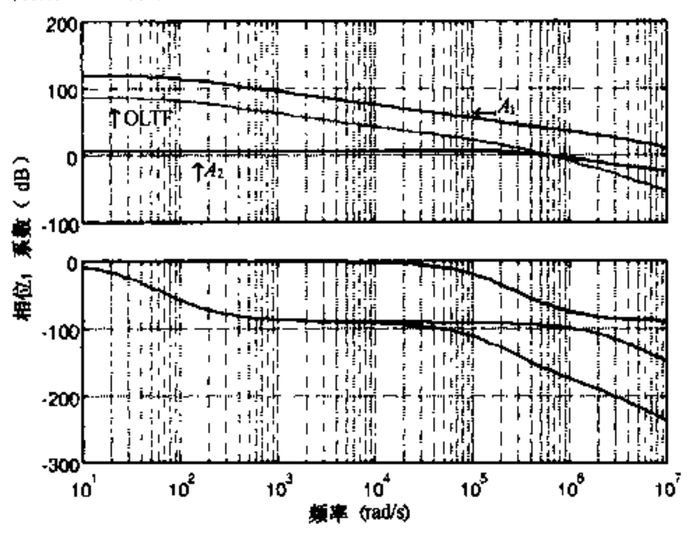


图 1-9　开环传递函数图

利用 margin 函数，可以获得相位裕度、增益裕度和相应的频率。

```
% Gain and phase margins of the OLTF

[mg,mp,wg,wp] = margin(sys)
disp(['Gain margin in dB = 'num2str(mg)])
disp(['Corresponding frequency = 'num2str(wg/(2*pi))])
disp(['Phase margin in degrees = 'num2str(mp)])
disp(['Corresponding frequency = 'num2str(wp/(2*pi))])

Gain margain in dB = 4.485
Corresponding Frequency = 207591.659
Phase margin in degress = 18.8604
Corresponding Frequency = 95346.57
```

相位裕度在 f_p=99.6 kHz 的 0.06°至 f_p=95.3 kHz 的 18.9°之间通过。这不足以确保控制系统的稳定性。文件 ampli2.m 可得到系统单位阶跃响应图。

由图 1-10 可以看到，稳定性的提高仍很低，输出振荡在稳定前后的 50μs 仍很剧烈。

在 $\dfrac{Vs}{V_1}$ 的关系下，晶体管级放大倍数实际上为 2.3，因为 A_2B_2 总是小于 1。为得到一个附加相位裕量，常采用在直链尾部（在功率层输出）加滞后校正的技术。当然，这是一个无源校正，形成放大器 R_s 输出阻抗的一部分和连在放大器输出的 R、C 串行网络的另一部分。

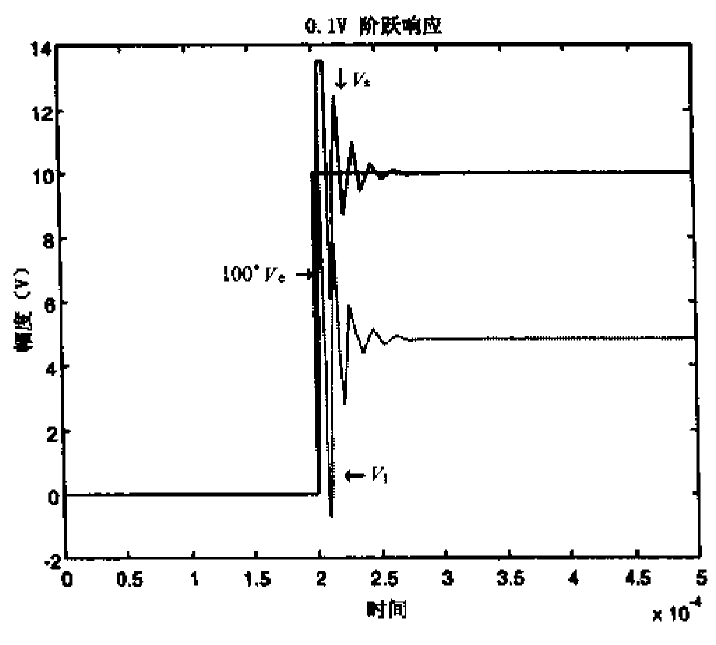


图 1-10　电压幅度响应图

1.4　相位滞后校正放大器

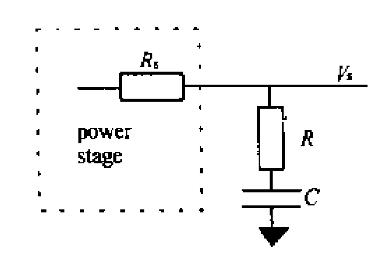


图 1-11　相位滞后校正放大器图

这样的相位滞后校正能被下面的 C（p）传递函数所描述：

$$C(p)=\frac{1+RCp}{1+(R+R_S)Cp}=\frac{1+\tau p}{1+a\tau p}$$ （其中 $a>1$）

修正后的方框图如图 1-12 所示。

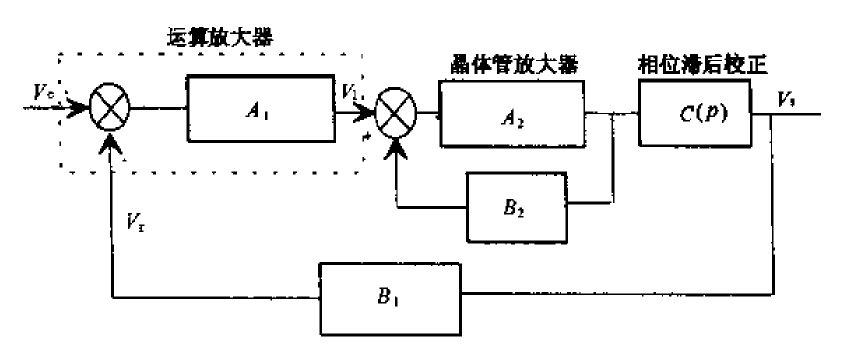


图 1-12　修正后的方框图

校正器有 2 个截止频率：

$$f_{c_1}=\frac{1}{2\pi a\tau}$$

$$f_{c_2}=\frac{1}{2\pi\tau}$$

我们应选择 f_{c_1} 和 f_{c_2}，这样最大相位延迟远在临界频率前发生，因为如果相位延迟在临界频率区将有助于减少相位裕度，这与期望的目的相反。

考虑大约降低增益 10dB。已知

$$|C(p)|_{\min}=20\log\left(\frac{1}{a}\right)$$

得到 a≈3，并且

$$\phi_M=-\arcsin\left(\frac{a-1}{a+1}\right)$$

对应于最大滞后相位 30°。

此外，在前面的模拟中，开环传递函数系数为 1 的临界频率在 207 kHz 下计算。这对于在 f_m=4 kHz 远小于临界频率下的最大相位延迟的固定非常重要。

$$f_m=\frac{1}{2\pi RC\sqrt{a}}$$

也就是

$$\tau=RC=2.25\times10^{-5}\,\mathrm{s}$$

文件 ampli4.m 可检查修正后的系统稳定性。

ampli4.m file

```
% Power amplifier
% Transistors stage feedback
% Phase lag corrector

% Open loop transfer function
% Operational amplifier stage
A1=1e6;
T1=0.0159;
T2=1.59e-7;
num1=A1;
den1=[T1*T2 T1+T2 1];
sys1=tf(num1,den1);

% Transistors stage
A2=10;
T3=1.59e-5;
B2=0.33;
num2=A2;
den2=[T3 1];
sys2=tf(num2,den2);
sys2b=sys2/(1+sys2*B2);
```

◆ 相位滞后校正传递函数

```
% Phase lag corrector
T=2.25e-5;
```

```
a=3;
num3=[T 1];
den3=[a*T 1];
sys3=tf(num3,den3);

Transfer function:
                        2.25e-005 s + 1
                        ---------------
                        6.75e-005 s + 1
```

◆ 整个放大器的开环传递函数

```
% Open loop transfer function
B1=0.01;
sys=sys1*sys2b*B1;
sysc=sys1*sys2b*sys3*B1;
[z,p,k]=zpkdata(sys);
sys_k=zpk(z,p,k)

Zero/pole/gain:
                   2.487761e018 (s+6.289e004)
  ----------------------------------------------------------
  (s+6.289e006) (s+2.704e005) (s+6.289e004) (s+62.89)
```

◆ $C(p)$校正和 OLTF 整个放大器的开环传递函数预测图

```
% Graph of the C(p) and OLTF transfer functions
figure(1)
w = logspace(1 ,7,100)
bode(sys3,w,'b'), hold on
bode(sysc,w,'r'), hold off
title('C(p) and OLTF transfer functions')
gtext('C(p)'), gtext('OLTF')
```

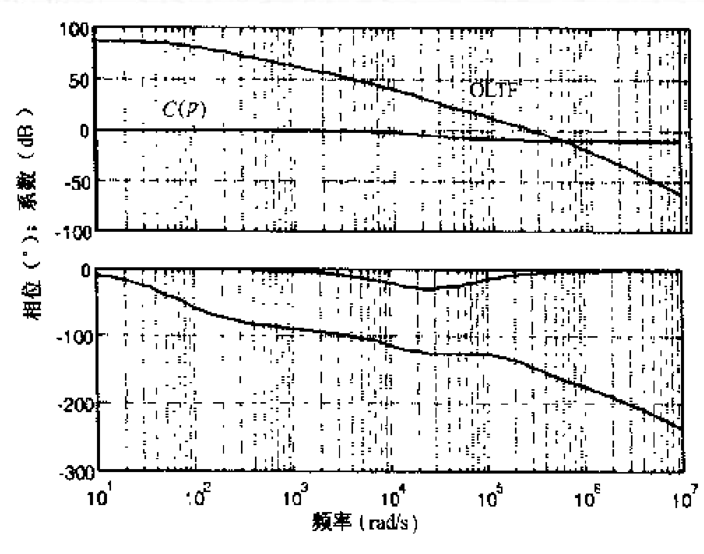


图 1-13　C（p）和 OLTF 传递函数

注意：在开环传递函数 0 dB 通道前，减少 10 dB 增益变得可实现。最大相位滞后在大约 4kHz 下，为 30° 左右。

```
% OLTF gain and phase margins
```

```
disp(['Gain margin in dB = 'num2str(mg)])
disp(['Corresponding frequency = 'num2str(wg/(2*pi))])
disp(['Phase margin in degrees = 'num2str(mp)])
disp(['Corresponding frequency = 'num2str(wp/(2*pi))])

Gain margain in dB = 11.9177
Corresponding Frequency = 195379.5866
Phase margin in degress = 32.1984
Corresponding Frequency = 50628.1479
```

修正后的系统相位裕度提高到32°，其SIMULINK模型如图1-14所示。为最佳地逼近真实系统，还必须考虑技术文档中有“slew rate”特性的运算放大器和晶体管功率级的上升和下降时间。“Rate limiter”包可提供此功能的安装。

一般的，“slew rate”取下列值：

- 运算放大器1 V/μs；
- 功率级0.1 V/μs。

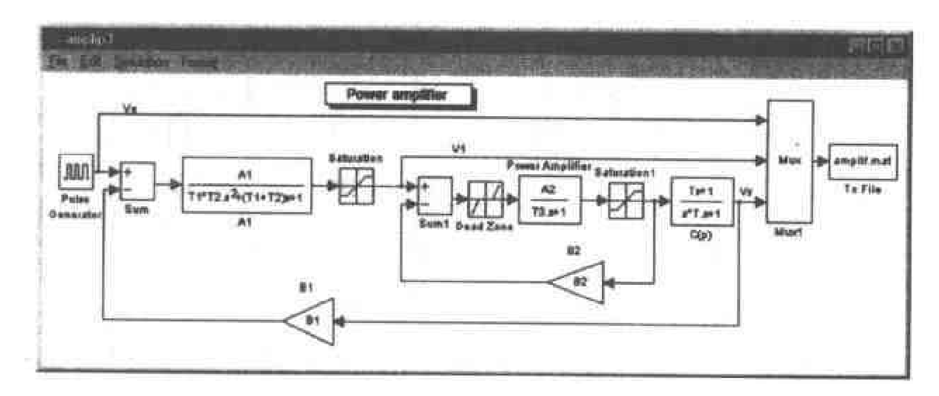


图1-14 放大器模型图

文件ampli2.mat可获得系统单位阶跃响应图。

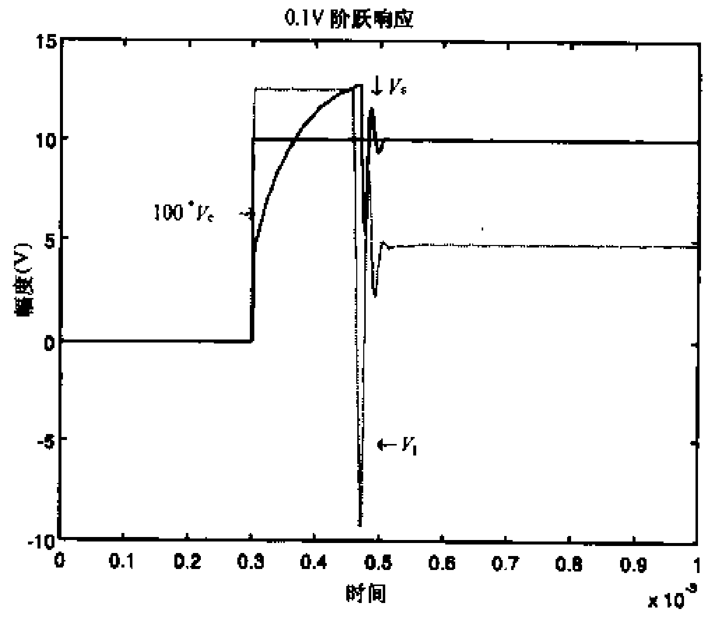


图1-15 系统阶跃响应图

取得的结果并不是理想的，因为在控制系统主环路中的校正产生了延迟。这不仅产生了输出信号的延迟，还通过环路在运算放大器输出产生很大的振荡。为了降低延迟影响，有必要减小τ而不许 f_m 太接近临界频率。因此必须互相折衷，但这并不是总有可能实现。为了补偿系统，考虑最后一种可能性。它由整个放大器产生的反馈（不是通过一个简单阻抗桥固定整个增益，而是通过超前校正）组成。这个微分作用在提高稳定性的同时提高了速度。

1.5 超前相位校正反馈放大器

原理（见图 1-16）

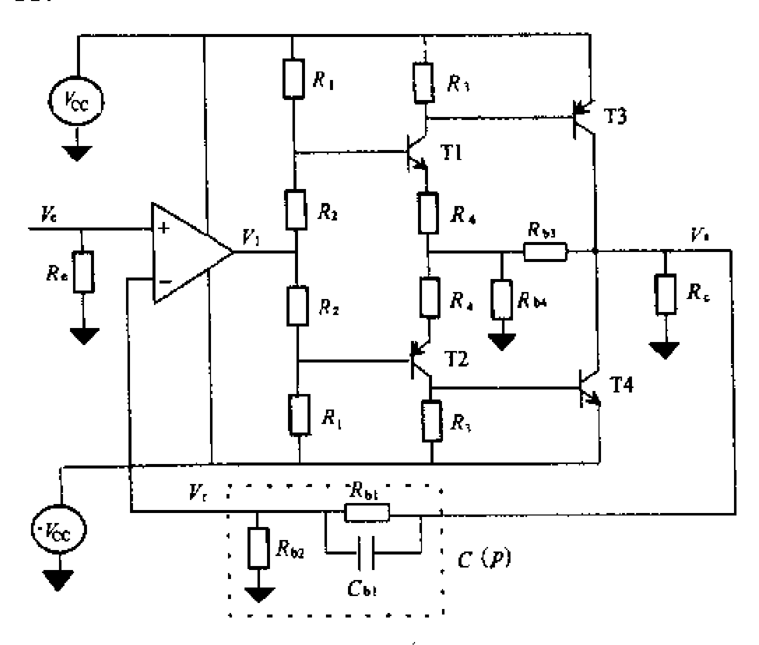


图 1-16　超前相位校正反馈放大器原理图

C（p）加在整个放大器的反馈上：

$$C(p)=K\frac{1+\tau p}{1+K\tau p}$$

其中：

$$K=\frac{R_{b2}}{R_{b1}+R_{b2}}$$

$$\tau=R_{b1}\times C_{b1}$$

固定静态增益（稳定状态）K 以及起远离放大器的工作频率的微分作用 τ，可使干扰只在瞬间发生。如使放大器放大倍数为 100，通带为 20 kHz，则

$$K=0.01$$

$$\tau=\frac{1}{4\pi BP}=4\times10^{-6}$$

这导致如图 1-17 所示的 SIMULINK 模型。

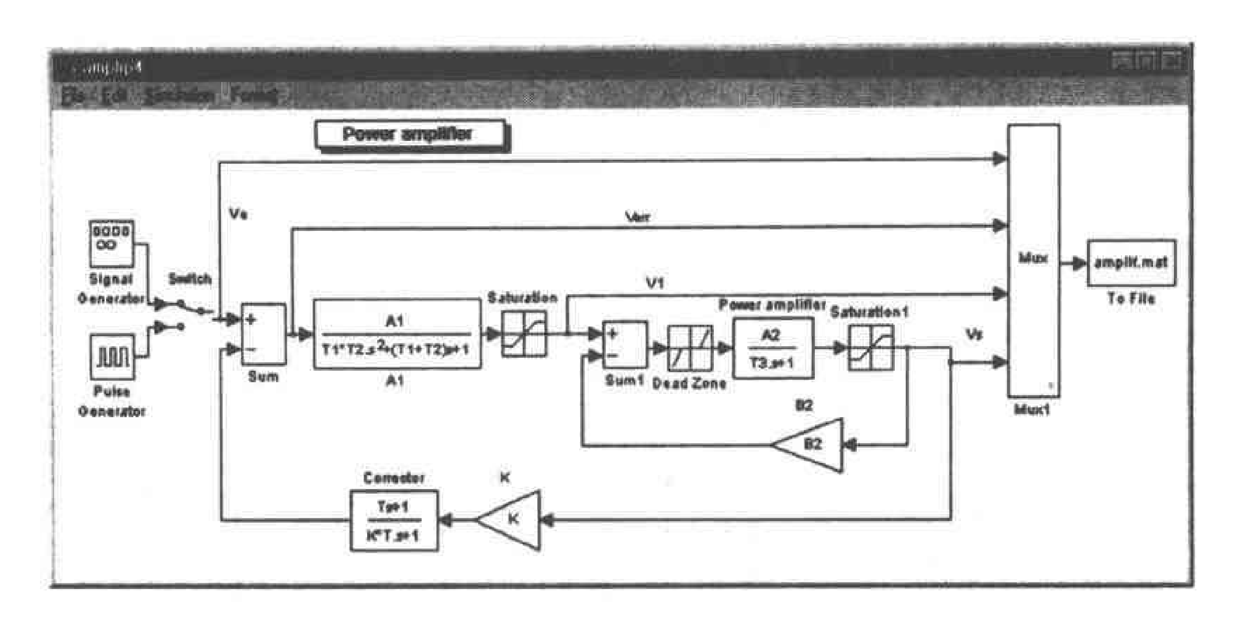


图 1-17　SIMULINK 功率放大器模型

修正后的放大器在 0.1 V 幅度的阶跃和 10 kHz 频率的正弦信号下的阶跃和谐波响应如图 1-18 所示。

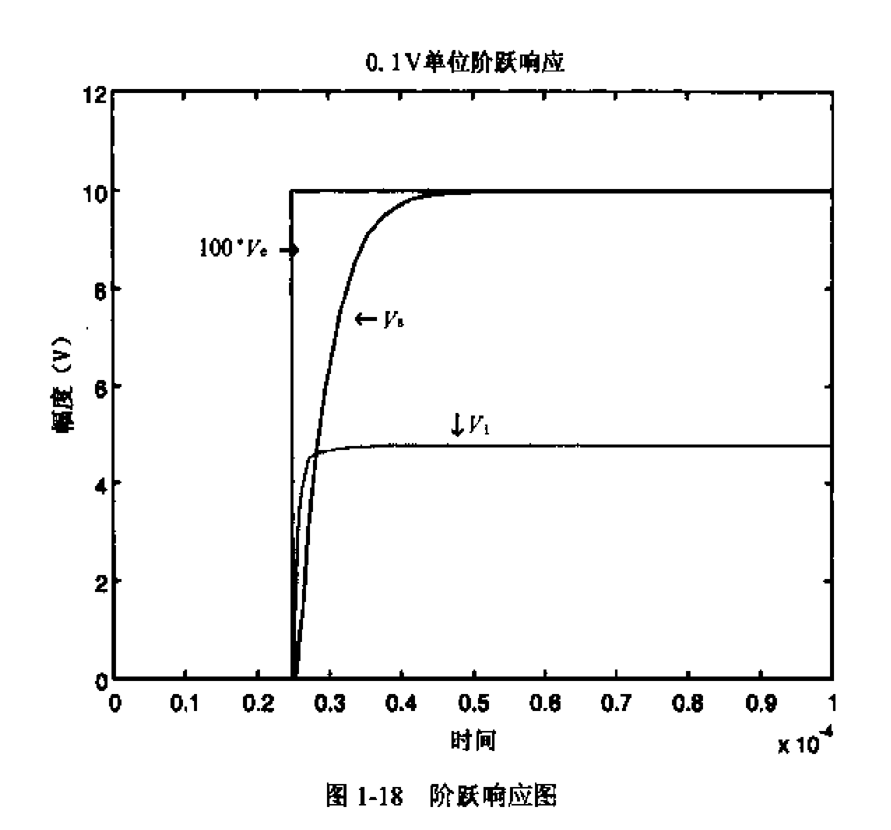


图 1-18　阶跃响应图

0.1V 阶跃无超调响应需要 15μs。

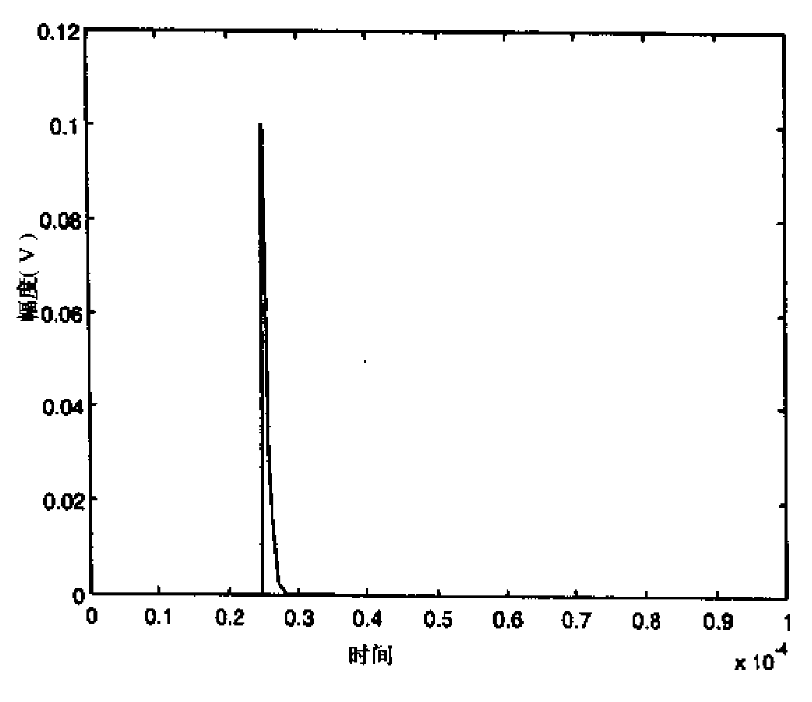


图 1-19　误差信号图

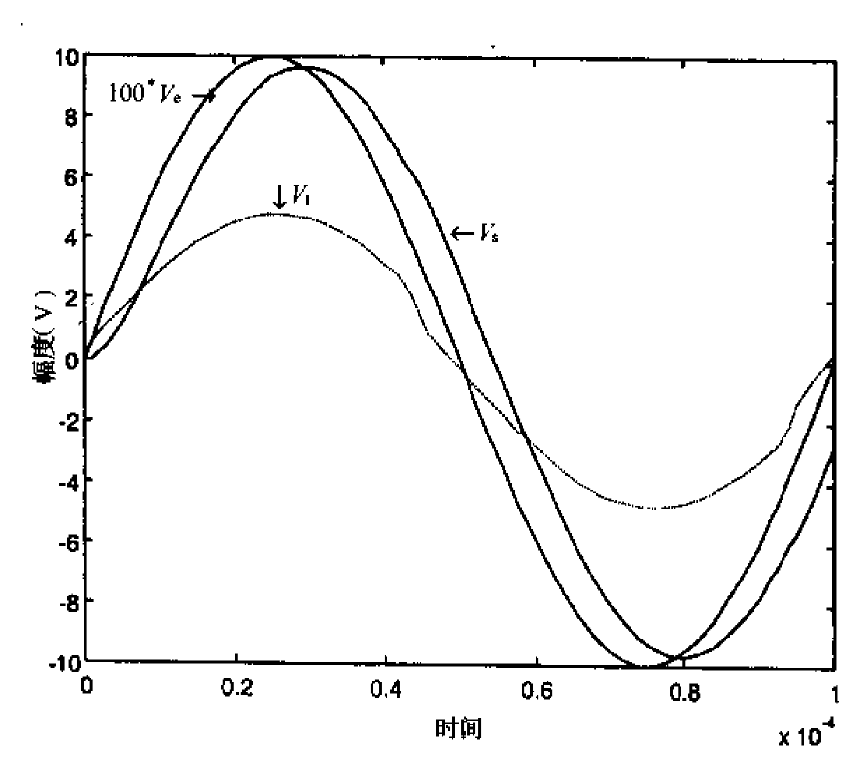


图 1-20　谐波响应图

可以看到，由于微分校正作用，放大倍数低于 100。在 V_1 信号而不是输出处明显出现一死区作用，这是因为反馈削弱了整体的失真。

应用2 电磁悬浮

这个应用是通过使用电磁能让块M在无任何接触下移动（见图2-1）。

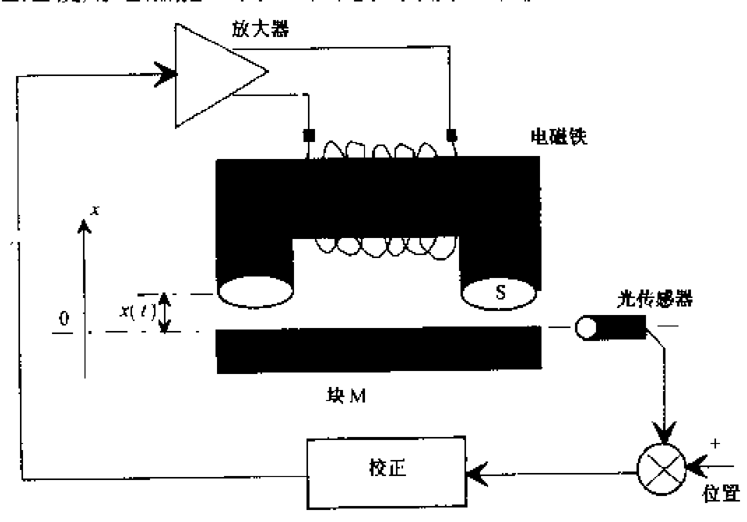


图2-1 组成原理图

这个控制系统包括距电磁铁x_0距离的块M移动部分。

- $M = 1\ \text{kg}$ 移动块质量；
- $S = 4\ \text{cm}^2$ 电磁铁表面；
- $N = 1\ 000$ 电磁线圈的圈数；
- $r = 2\ \Omega$ 电磁线圈电阻；
- $x_0 = 5\ \text{mm}$ 移动块和电磁铁之间的控制距离。

空气磁导率μ_0等于$4\pi\times10^{-7}$，电磁线圈和移动块的磁材料的磁导率可看做非常大。

2.1 过程模型

2.1.1 用线圈电流I和气隙e表示的吸引力F表达式

磁路的磁阻由下列关系表示：

$$R = \int_e \frac{\mathrm{d}l}{\mu_0 \mathrm{d}s} + \int_e \frac{\mathrm{d}l}{\mu_0 \mu_r \mathrm{d}s}$$

已知$\mu_r \to \infty$，得

$$R = \frac{2e}{\mu_0 S}$$

电磁感应可写为

$$B = \frac{\phi}{S\cos\varphi}$$

这里φ是磁力线和表面 S 垂线间的夹角。如果 $\varphi = 0$，得到

$$B = \frac{\phi}{S}$$

引力为

$$F = \frac{B^2 S}{\mu_0}$$

此外，由 $NI = R\phi$，得到

$$B = \frac{\phi}{S} = \frac{NI}{RS}$$

$$F = \frac{B^2 S}{\mu_0} = \left(\frac{NI}{RS}\right)^2 \frac{S}{\mu_0} = \frac{\mu_0 S N^2 I^2}{4e^2}$$

2.1.2 工作点 $e(t)=e_0$ 附近过程的线性化

引力 F 由线圈电流 $I(t)$和气隙距离 $e(t)$这两个变量决定。通过假定悬浮块移动的距离与 e_0 相比仍很小来限制这个作用。所考虑的系统能简化成由如下的近似公式描述的一个线性系统：

$$F(x, I) = F(x = e_0, I = I_0) + k_1 I(t) + k_2 x(t)$$

已知 $F = Mg$，悬在距离 $x = e_0$ 下的物块电流表达式为

$$I_0 = \frac{2e_0}{N}\sqrt{\frac{Mg}{\mu_0 S}}$$

系数 k_1 和 k_2 表达成

$$k_1 = \frac{dF}{dI} = \frac{d}{dI}\left(\frac{\mu_0 S N^2 I^2}{4{e_0}^2}\right) = \frac{\mu_0 S I_0 N^2}{2{e_0}^2}$$

$$k_2 = \frac{dF}{dx} = -\frac{dF}{de} = -\frac{d}{de}\left(\frac{\mu_0 S N^2 I^2}{4{e_0}^2}\right) = \frac{\mu_0 S I_0^2 N^2}{2{e_0}^3}$$

由牛顿第一定律，得到表达式

$$\sum F = M\gamma$$

$$k_1 I(t) + k_2 x(t) = M\frac{d^2 x}{dt^2}$$

2.1.3 过程传递函数

因为电磁可由电流或电压控制，所以可写出位移 $x(t)$、电流 $I(t)$和放大器输出电压 $U(t)$之间的传递函数表达式。

◆ 电磁电流控制

$$k_1 I(t) + k_2 x(t) = M\frac{d^2 x(t)}{dt^2}$$

此方程的拉普拉斯变换为：

$$k_1 I(p) + k_2 X(p) = M p^2 X(p)$$

也就是

$$\frac{X(p)}{I(p)}=\frac{k_1}{M}\times\frac{1}{p^2-\frac{k_2}{M}}=\frac{b_0}{p^2+a_0}$$

这里

$$\begin{cases}b_0=\dfrac{k_1}{M}\\ a_0=\dfrac{-k_2}{M}\end{cases}$$

◆　电磁电压控制

$$U(t)=rI(t)+\frac{\mathrm{d}\phi(t)}{\mathrm{d}t}$$

其中

$$\begin{cases}\phi(t)=L(t)I(t)\\ L(t)=L_0-\dfrac{\mathrm{d}L}{\mathrm{d}x}\end{cases}$$

给出一个电压控制表达式

$$U(t)=rI(t)+L_0\frac{\mathrm{d}I(t)}{\mathrm{d}t}-I_0\frac{\mathrm{d}L(t)}{\mathrm{d}x}\times\frac{\mathrm{d}x(t)}{\mathrm{d}t}$$

其中

$$L=\frac{N\phi}{I}=\frac{N^2}{R}=\frac{\mu_0N^2S}{2e}$$

$$\frac{\mathrm{d}L}{\mathrm{d}x}=-\frac{\mathrm{d}L}{\mathrm{d}e}=-\frac{\mu_0N^2S}{2e}$$

也就是

$$U(t)=rI(t)+L_0\frac{\mathrm{d}I(t)}{\mathrm{d}t}+\frac{\mu_0N^2SI_0}{2e^2}\times\frac{\mathrm{d}x(t)}{\mathrm{d}t}$$

因此

$$U(t)=rI(t)+L_0\frac{\mathrm{d}I(t)}{\mathrm{d}t}+k_1\frac{\mathrm{d}x(t)}{\mathrm{d}t}$$

上式的拉普拉斯变换为

$$U(p)=rI(p)+L_0pI(p)+k_1pX(p)$$

已知

$$\frac{X(p)}{I(p)}=\frac{k_1}{M}\times\frac{1}{p^2-\frac{k_2}{M}}$$

获得如下位移和电磁控制电压之间的传递函数公式:

$$\frac{X(P)}{U(p)}=\frac{k_1}{k_1^2p+(r+L_0p)(Mp^2-k_2)}=\frac{b_0}{a_2p^3+a_2p^2+a_1p+a_0}$$

这里

$$\begin{cases} b_0 = k_1 \\ a_3 = L_0 M \\ a_2 = rM \\ a_1 = k_1^2 - k_2 L_0 \\ a_0 = -rk_2 \end{cases}$$

选取

$$\begin{cases} k_1 \approx 14 \\ k_2 \approx 3941 \\ L_0 \approx 50\,\text{mH} \\ I_0 \approx 1.4\,\text{A} \end{cases}$$

2.2 电流放大器控制系统

电磁吸引过程本质上是不稳定的。为克服不稳定，线圈中的电流必须在位置偏差 $x(t)$=0 时能很块地校正。因此，在控制位置 $x(t)$前，必须控制电磁线圈的电流 $I(t)$。

◆ $I(t)$电流控制系统简介（见图 2-2）

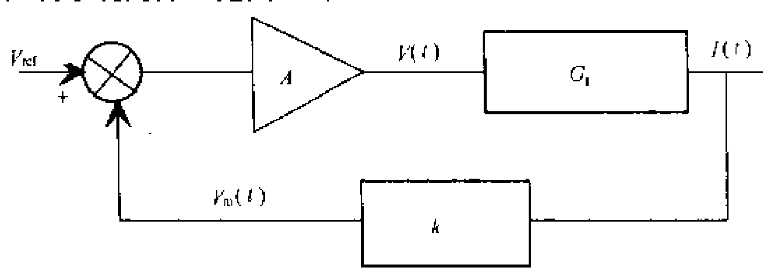


图 2-2 电流控制系统方框图

考虑感应系数 L 等于常数 L_0。线圈电流由 $V_m(t)$电压的转换系数 k 等于 0.04 下的增益产生：

$$G_1(p) = \frac{I(p)}{V(p)} = \frac{1}{r + L_0 p} = \frac{K}{1 + \tau p} = \frac{0.5}{1 + 0.025p}$$

为使 I_0 等于 1.4 A，有必要假设 V_{ref}=56 mV 作为参考。闭环传递公式如下：

$$\frac{V_m(p)}{V_{ref}(p)} = \frac{kKA}{1 + kKA} \times \frac{1}{1 + \dfrac{\tau}{1 + kKA} p}$$

如果制定一条绝对规则来降低过程时间常数至 5 ms，增益 A 须取值 200（见图 2-3）。

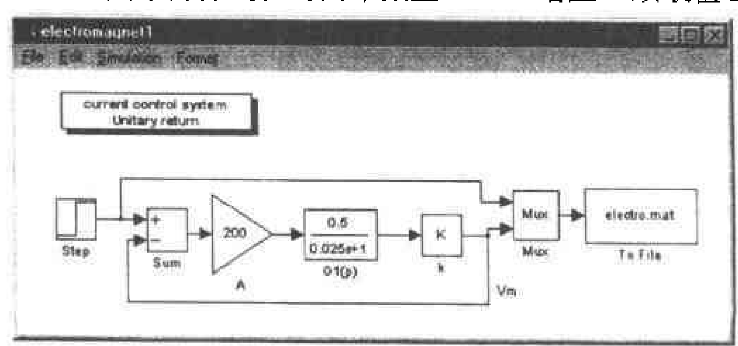


图 2-3 电流控制系统单一反馈图

◆ 56 mV 阶跃响应

从阶跃响应模拟图 2-4 可以看到，对应于达到最终电压 V_1 的 63%的一阶控制系统的时间常数为 5 ms。

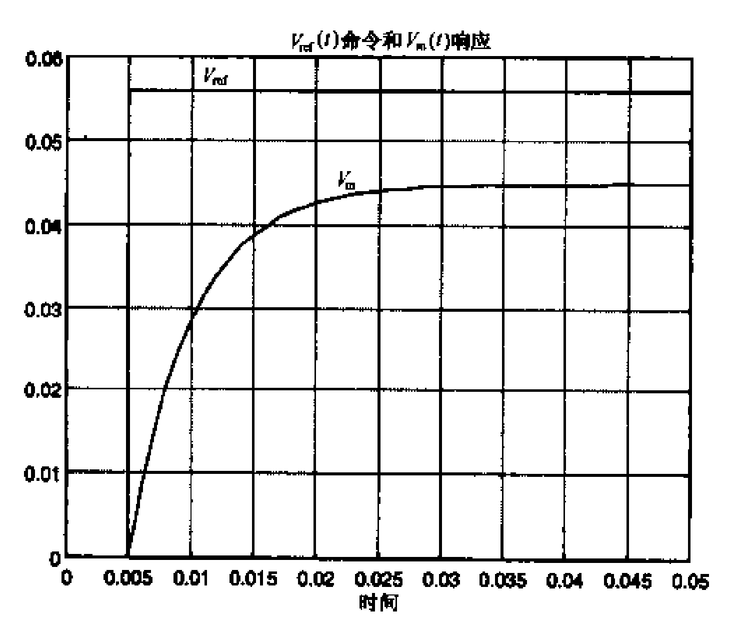


图 2-4 阶跃响应图

另一方面，位置误差也很重要。为在不求助很重要的放大值 A 的前提下减小误差，可以考虑使用比例和积分校正。为此，在固定微分校正为 0 时使用类似 PID 的 SIMULINK 块（见图 2-5）。

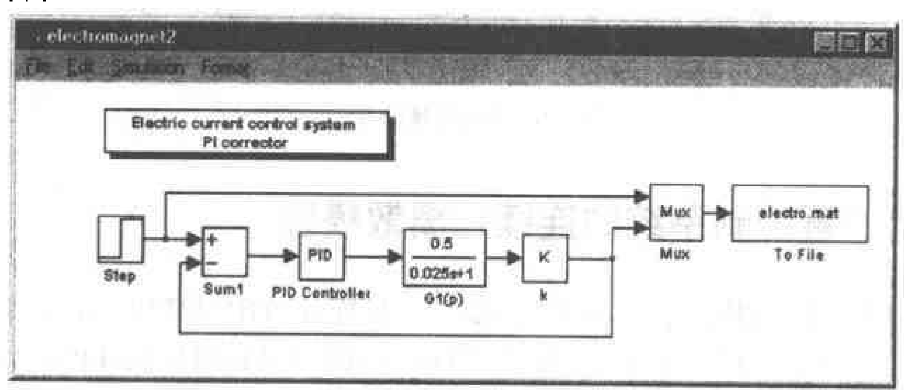


图 2-5 电流控制系统 PI 控制器

PI 校正关系为

$$C(p) = A + \frac{I}{p}$$

因此，控制系统传递函数写为

$$\frac{V_m(p)}{V_{ref}(p)} = \frac{1 + \frac{A}{I}p}{1 + \frac{(1+kKA)}{kKI}p + \frac{\tau}{kKI}p^2}$$

此表达式的自然振荡角频率和阻尼系数为

$$\begin{cases}\omega_0 = \sqrt{\dfrac{kKI}{\tau}} \\ \xi = \dfrac{1}{2} \times \dfrac{1+kKA}{\sqrt{kKI\tau}}\end{cases}$$

如果制定一条绝对规则以获得自然角频率为 500 rad/s 和阻尼系数ξ =0.7，则获得校正环节的参数

$$\begin{cases}A = 825 \\ I = 312\,500\end{cases}$$

◆ 控制系统阶跃响应（见图 2-6）

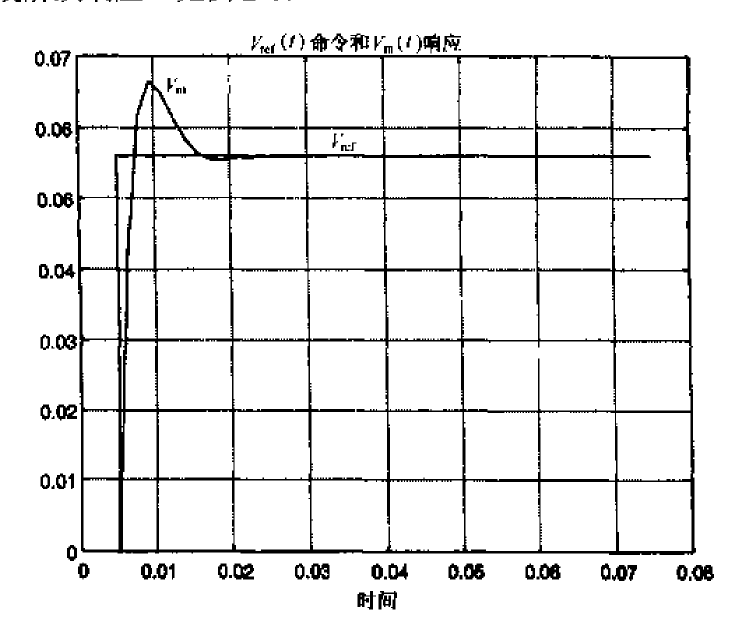
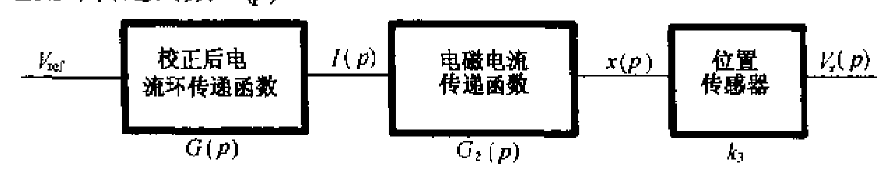


图 2-6　控制系统阶跃响应

2.3　x(t)位置控制系统的连续和离散模型

将电流控制系统集成在位置 $x(t)$控制回路中。位置 $x(t)$的传感器假定在点 x=0 附近为线性，产生一个位置电压函数 $V_x(t)$。灵敏度 1 V/mm 对应于转换增益 k_3=1 000 V/m。注意上述的校正后电流环传递函数 $G(p)$。



上图所示的前向传递函数为：

$$\frac{V_x(p)}{V_{ref}(p)} = \frac{k_1 k_3}{kM} \times \frac{1+\dfrac{A}{I}p}{\left(1+\dfrac{1+kKA}{kKI}p+\dfrac{\tau}{kKI}p^2\right)\left(p^2-\dfrac{k_2}{M}\right)}$$

下面的程序可计算出上面 3 个方框的传递函数。

```
% Electromagnet mass and surface
M = 3; S = 4e-4;
% Turns number and coil resistance
N = 1000; r = 2;
% Average current in the bobbin
I = 1.4;
% Air gap width and mu constant
e = 0.005; mu = pi*4e-7;
% Gain terms
k = 0.04; k3 = 1000;
% PI corrector
A = 825; Int = 312500;
% Ol transfer fonction
K = 0.5; tau = 0.025;
k1 = mu*S*I*N^2/(2*e^2);
k2 = mu*S*(N*I)^2/(2*e^3);
```

◆ 传递函数 $X(p)/I(p)$表达式

```
num1 = k1/M; den1 = [1 0 -k2/M];
sys1 = tf(num1,den1)
Transfer function
14.07
-----------
s^3-3941
```

◆ 传递函数 $V_m(p)/V_{ml}(p)$表达式

```
num2=[1+A/Int 0];
den2=[tau/(k*K*I) (1+k*K*A)/(k*K*Int) 1];
syn2=tf(num2,den2)

Transfer function
   1.003s
   ---------------------------------
   0.8929s^2+0.0029s+1
```

◆ 传递函数 $V_s(p)/V_{ml}(p)$表达式

```
sys=sys1*syn2/k

Transfer function
            352.8s
----------------------------------------------------------------
0.8929s^4+0.0028s^3-3518s^2-11.03s-3941
```

这样一个过程本质上是不稳定的。为检查传递函数的稳定性，把极点显示出来比较好。

```
% Writing in zeros-poles form
zsys=zpk(sys)

Zero/pole/gain:
         395.1218s
----------------------------------------------------------
(s-62.78)(s+62.78)(s^2+0.003136s+1.12)
```

显然，对于实数部分 p=62.78 的极点，系统不稳定。同样可用 pzmaps 极点和零点绘图命令来检查。

```
figure(1);
pzmap(sys),grid;
```

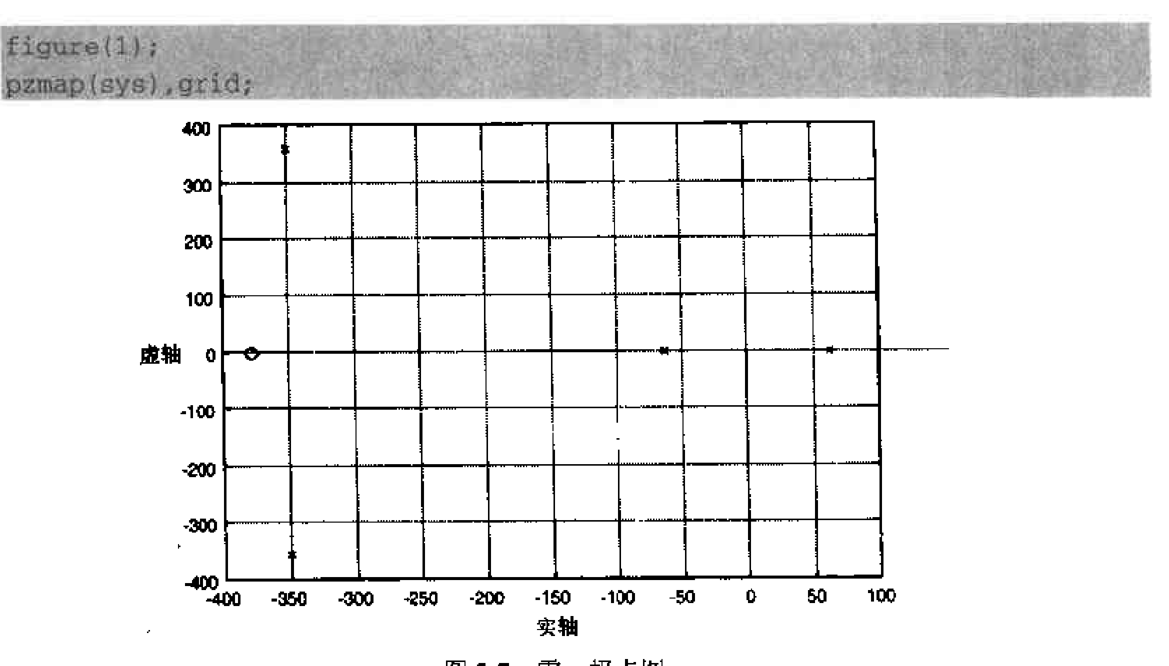


图 2-7 零、极点图

为了对 SIMULINK 模型进行位置控制，电流控制系统部分将封装成为子系统（见图 2-8）。

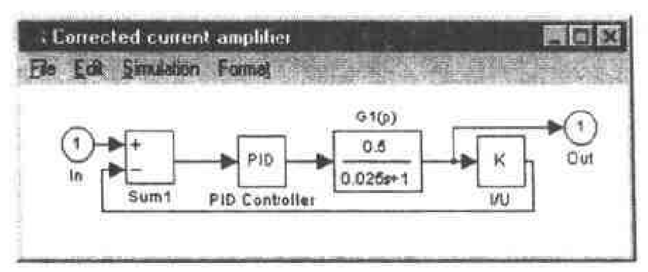


图 2-8 修正后的电流放大器

具有单位反馈的 SIMULINK 位置控制系统模型如图 2-9 所示。

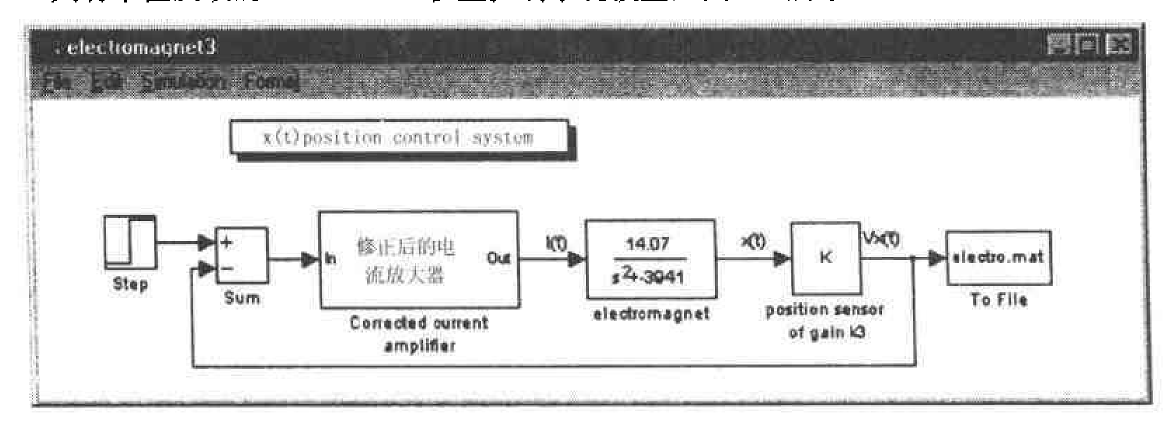


图 2-9 $x(t)$位置控制系统

在阶数不变的情况下的模拟结果证实了过程不稳定状态（见图 2-10）。

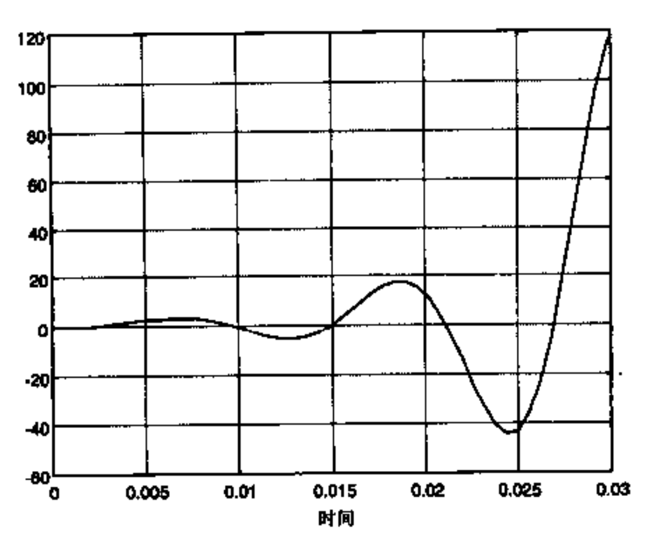


图 2-10 不变阶数下的 $V_x(t)$响应

在 c2dt 控制的帮助下可获得具有零阶保持器的$\frac{V_x}{V_{ref}}$前向通道离散模型。采样周期为 0.2 ms，大约是过程时间常数的 1/10。

```
% Discretisation by ZOH with Ts=0.2ms
Ts=0.2e-3;
sysd=c2d(sys,Ts,'zoh')
sysd1=zpk(sysd)

Transfer function
0.0003047 z^3+0.0009166 z^2-0.0008165 z-0.0002735
-------------------------------------------------
 z^4 - 3.86 z^3 + 5.59 z^2-3.599z+0.8694
Sampling time:0.0002
Zero/pole/gain:
0.00030465(z+3.672)(z-0.927)(z+0.2637)
------------------------------------------------------
(z-1.013)(z-0.9875)(z^2-1.86z+0.8694)
Sampling time:0.0002
```

2.4 x(t)数字随动控制

建议用零极点补偿法修正离散过程。开环过程 Z 传递函数为

$$T(z)=\frac{B(z)}{A(z)}=\frac{G(z+3.672)(z-0.927)(z+0.2637)}{(z-1.013)(z-0.9875)(z^2-1.86z+0.8694)}$$

它的分子表达式可变为

$$B(z)=B_1(z)B_2(z)$$

其中

$$\begin{cases}B_1(z)=(z+3.672)\\B_2(z)=G(z-0.927)(z+0.2637)\end{cases}$$

分母

$$A(z)=A_1(z)A_2(z)$$

其中

$$\begin{cases}A_1(z)=(z-1.103)(z^2-1.86z+0.8694)\\A_2(z)=(z-0.9875)\end{cases}$$

$B_1(z)$包含了不稳定的零点，因此不能被校正所补偿。$A_1(z)$含有不稳定的极点，这必须被校正所补偿。还有一对共轭复数极点也需补偿。因此，预期的校正后开环传递函数应有如下形式：

$$T_c(z)=\frac{B_c(z)}{A_c(z)}=\frac{KB_1(z)}{A_2(z)(z-1)}$$

项（z-1）不仅能取消位置误差，还能得到分母幂次必须大于等于分子幂次这一关系。增益k是调节参数。确保B_1和A_2多项式补偿而引入的校正环节可写为：

$$C(z)=\frac{KA_1(z)}{B_2(z)(z-1)}$$

定义了校正后，还必须计算调节参数的值，才能获得一个稳定的校正过程和满意的动态特性。过程的闭环传递函数为：

$$\text{FTBF}=\frac{KB_1(z)}{KB_1(z)+A_2(z)(z-1)}$$

如果极点在单位圆内，则系统稳定。

CLTF 分母可写为：

$$D=z^2+(K-1.9875)z+0.9875+3.672K$$

也就是

$$\Delta=K^2-18.663K+1.5625\times10^{-4}$$

还可写为

$$\Delta=(K-18.663)(K-8.3725\times10^{-6})$$

考虑 3 种情况：

- $\Delta=0$时有 1 个不稳定的重极点；
- $\Delta>0$时有 2 个不稳定的实数极点；
- $\Delta<0$时有 2 个稳定的共轭复数极点。

$$z_{1,2}=\frac{1.9875-K\pm \mathrm{j}\sqrt{-K^2+18.663K-1.5625\times10^{-4}}}{2}$$

如果$8.37\times10^{-6}>K>3.4\times10^{-3}$，则导致$|z_{1,2}|<1$。

极点为共轭复数，闭环系统的特征式为如下类型：

$$D(z)=z^2+\alpha z+\beta$$

其中：

$$\alpha=-2\exp(-\xi\omega_0T)\cos(\omega_0T\sqrt{1-\xi^2})$$
$$\beta=\exp(-2\xi\omega_0T)$$

记$\omega_0T=x$，根据 CLTF 分母表达式，可得到试图确定的具有同样增益K的 2 个表达式：

$$K = 1.9875 - 2e^{-\xi x}\cos(x\sqrt{1-\xi^2}) = f(x)$$

$$K = \frac{e^{-2\xi x} - 0.9875}{3.672} = g(x)$$

让阻尼系数$\xi = \frac{\sqrt{2}}{2}$以得到最小响应时间。这里试图通过图解方法寻找增益 K 值。为此，在同一坐标系下画出$f(x)$和$g(x)$函数曲线。2 条曲线的交叉点的纵坐标为增益 K，横坐标为降低的固有频率$\omega_0 T$。

mass_lev.m file

```
clear all

x = 0:0.1:1;
z = sqrt(2)/2;
f = 1.9875-2*exp(-z*x).*cos(x*sqrt(1-z^2));
g = (exp(-2*z*x)-0.9875)/3.672;

% graph of the two curves
figure(1)
plot(x,f)
hold on
plot(x,g,':'), hold off
title('Graphic determination of the k gain')
xlabel('x = \omega0T')
text(0.3,-0.03,...
'\downarrow1.9875-2e^{-\zetax}cos[x(1-zeta^{2})^{1/2}]',...
'HorizontalAlignment','left')
text(0.53,0.7,...
    '\leftarrow (e^{-2 \zeta x} - 0.9875)/3.672)',...
    'HorizontalAlignment','left')
```

在K和$\omega_0 T$值小时，2 条曲线才相交。同时必须考虑减少的频率和采样周期（见图 2-11）。

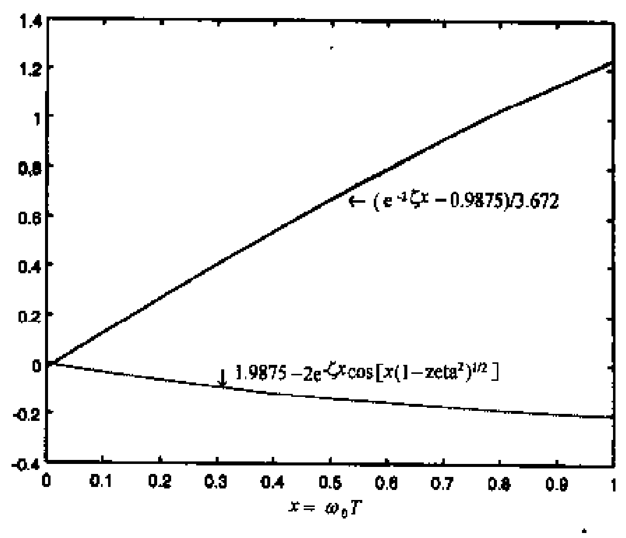


图 2-11　增益K的图形决定法

在很小采样周期下定义减少频率的间距。

```
x = 0:1e-6:0.02; g = (exp(-2*z*x)-0.9875)/3.672;
f = 1.9875-2*exp(-z*x).*cos(x*sqrt(1-z^2));
plot(x,f), hold on, plot(x,g,':')
title('Graphic determination of the k gain')
axis([0 0.02 -0.005 0.005]), xlabel('x = \omega0T')
```

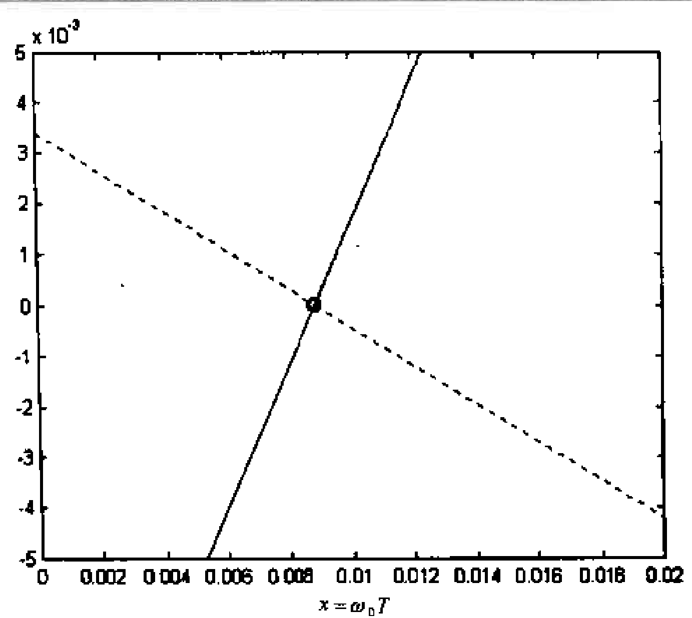


图 2-12　增益 K 的图形决定法

寻找使 f 和 g 值阵列拥有接近 10^{-6} 的同样值的指标。

```
i = find(abs(f-g)<0.000001)
h = plot(x(i),f(i),'o'); set(h,'LineWidth',2)
```

在指定精度下得到增益值 K。

```
f(i)
ans=
1.7041e-005

g(i)
ans=
  1.6560e-005
```

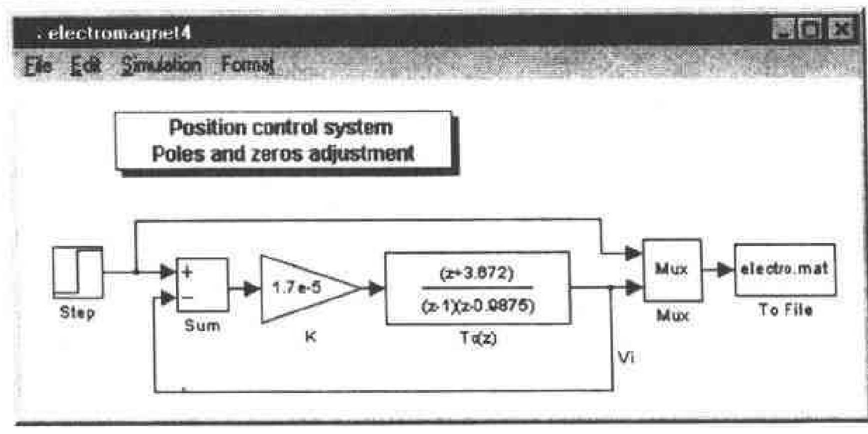


图 2-13　位置控制系统零、极点调节图

下面的命令可读 electro.mat 文件且得到阶跃响应图（见图 2-14），SIMULINK 在此文件中存储了时间、阶次和输出。

```
% reading of the simulation results matrix
load electro.mat
t = signal(1,:);
y = signal(3,:);
r = signal(2,:);

% Graph of the step response
plot(t,r)
hold on
grid
h = plot(t,y)
set(h,'LineWidth',1.5)
xlabel('Time in seconds')
title(' Position step follow-up')
```

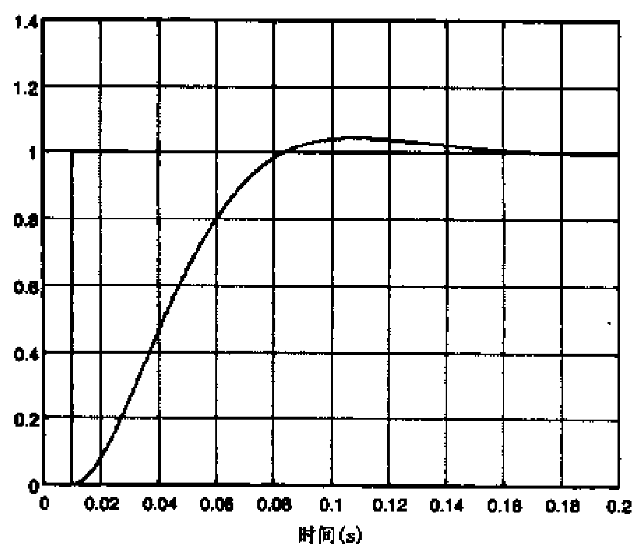


图 2-14 位置阶跃响应图

2.5 使用模糊调节器

模糊调节器的主要优点就是它的鲁棒性，这在控制本质不稳定的过程中尤其重要。

模糊控制包括 2 个输入，$e(t)$表示位置指令和测量值间的误差，$de(t)$是 $e(t)$的微分。控制输出称为 V_c。

3 个信号取值范围如下：

$$-1 < e < 1$$

$$-100 < de < 100$$

$$-1 < V_c < 1$$

2.5.1 变量模糊化

定义 3 个输入 trapmfs 梯形关系，neg 表示负，zero 表示零，pos 表示正。输出也定义 5 个梯形关系函数（GN 代表大的负值，N 为负，Z 为零，P 为正，GP 为大的正值）。

在图形界面帮助下得到的模糊修正称为 EA.fis。下面的命令呈现了这些模糊设置。

fuzzy1.m file

```
fismat = readfis('EA.fis');

figure(1)
plotmf(fismat,'input',1), grid
title('e(t) input membership rules')
xlabel('e(t) static error')
ylabel('Degree of membership')

figure(2)
plotmf(fismat,'input',2), grid
title('de(t) input membership rules')
xlabel('de(t)position error derivative')
ylabel('Degree of membership ')

figure(3)
plotmf(fismat,'output',1), grid
title('Vc(t) output membership rules')
xlabel('Vc(t) output')
ylabel('Degree of membership ')
```

◆ 输入“位置误差”的隶属函数（见图 2-15）

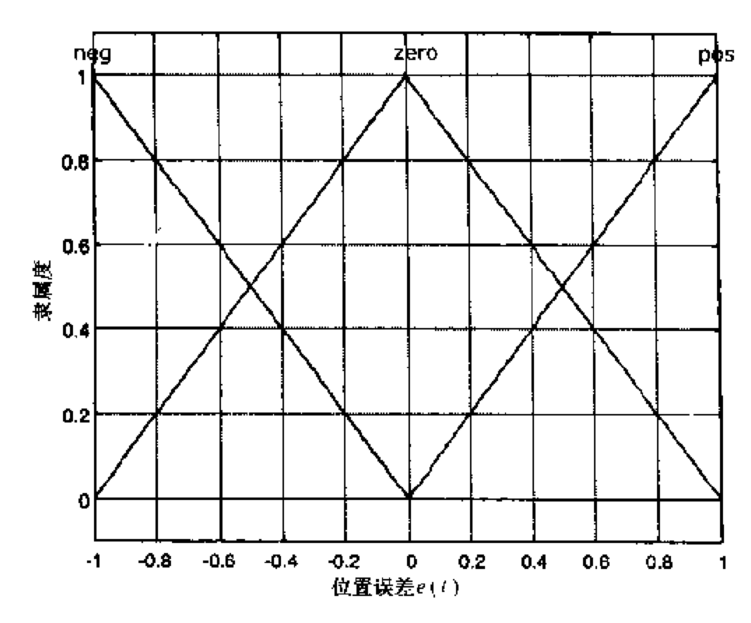


图 2-15 输入“位置误差”的隶属度图

◆ 输入“位置误差微分”的隶属函数（见图 2-16）

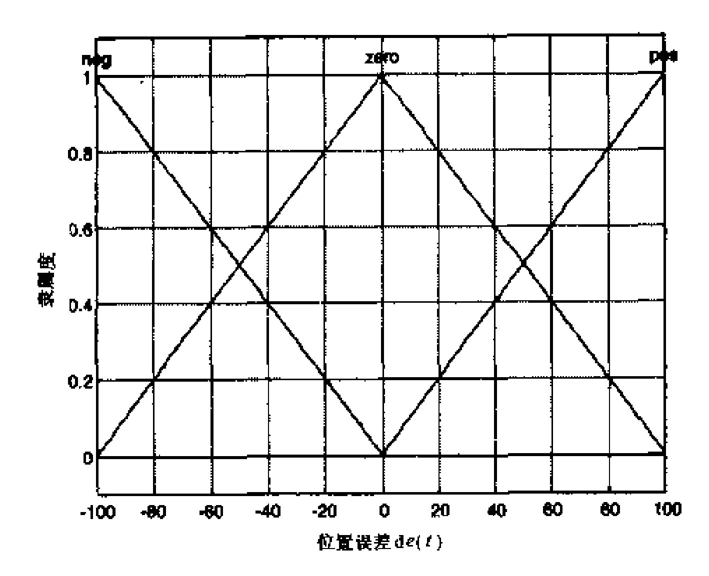


图 2-16　输入“位置误差微分”的隶属函数图

◆　输出“电磁控制”的隶属函数（见图 2-17）

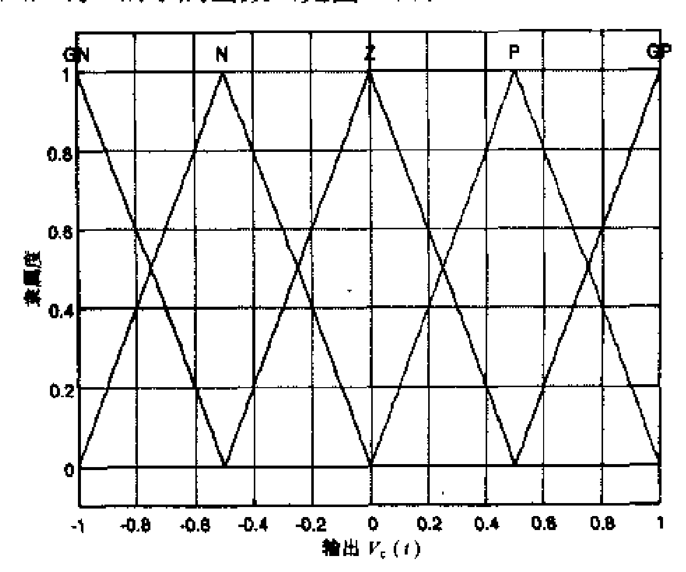


图 2-17　输出“电磁控制”的隶属函数图

使用最小算子 AND、最大算子 OR 的 Mamdani 方法。

2.5.2　推理规则定义

排列中有 2 个输入，每个输入有 3 个隶属规则，这就产生 9 种规则。9 种规则定义如下。

V_c

de \ e	NEG	ZERO	POS
NEG	GP	P	Z
ZERO	P	Z	N
POS	Z	N	GN

控制 ruleview 可使我们看到在冗长状态下编辑的规则（见图 2-18）。

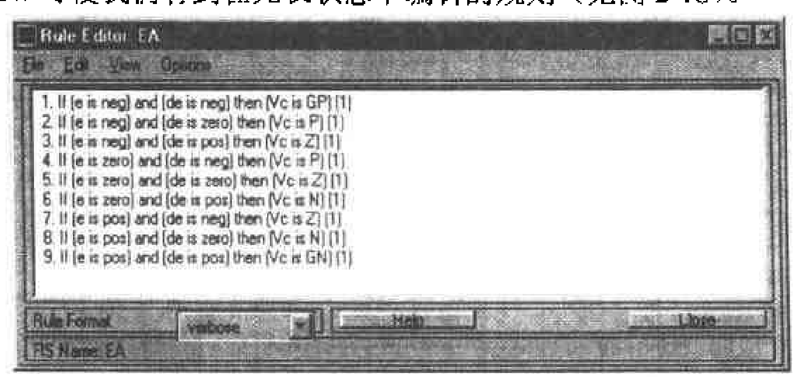


图 2-18　规则编辑器

2.5.3　输出解模糊

使用最常用的重心计算方法，选择它是因为准确性。控制 gensurf 能够根据输入 *e* 和 de 而绘制出输出变量 V_c 的表面图（见图 2-19）。

```
figure(4)
gensurf(fismat)
title('Fuzzy corrector control surface')
view(-110,-28)
```

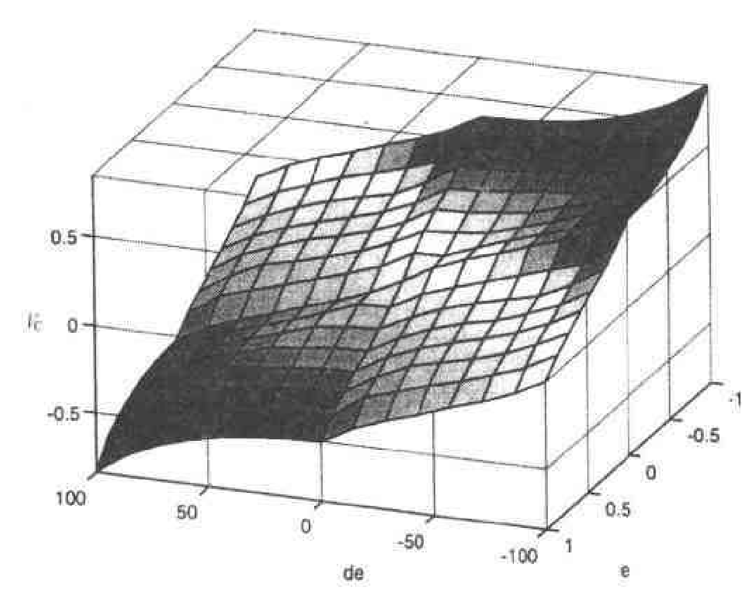
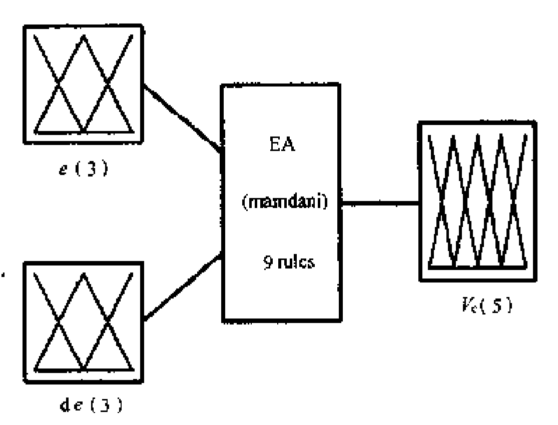


图 2-19　模糊控制器控制表面

在命令 plotfis 帮助下得到模糊系统的框图（见图 2-20）。

```
figure(5)
plotfis(fismat)
gtext(' Control system of the mass position')
```



EA 系统:2 输入,1 输出,9 法则

图 2-20　块位置控制系统

用零阶保持的离散化 SIMULINK 模型过程在图 2-21 模糊调节器的帮助下进行调节。

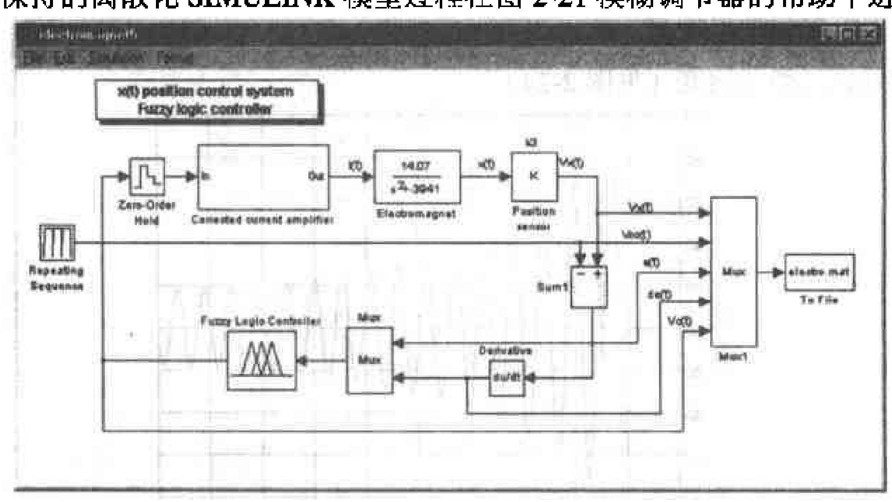


图 2-21　位置控制系统 $x(t)$模糊调节器

施加阶梯型指令，但为避免微分块太过剧烈的反应而呈现一定的转换斜面。

过程响应很快，振荡在指令位置处很小。振幅为 15 mV，使移动部分有 0.03 mm 的振荡。为减弱振荡，有必要在位置为 0 附近使过程有更大的衰减。为此，降低输出隶属规则 Z 的倾斜度（见图 2-22）。

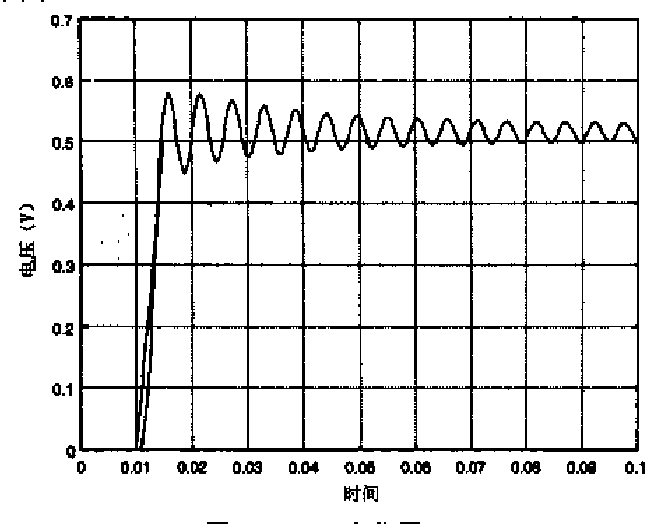


图 2-22　V_{xc}和位置 V_x

◆ 位置误差演变（见图 2-23）

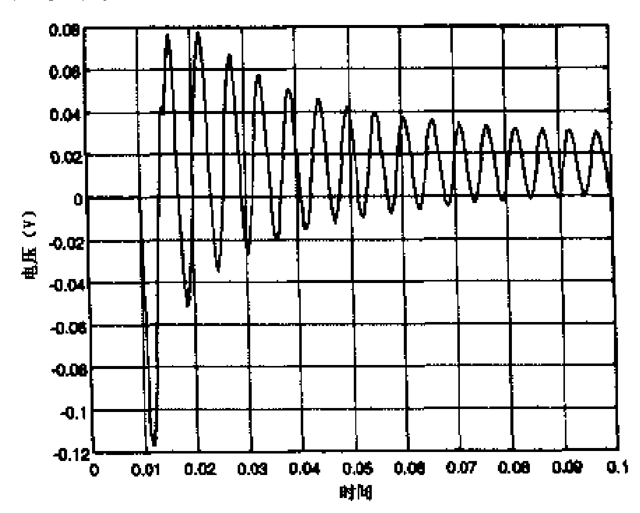


图 2-23 位置误差 $e(t)$

◆ 位置误差微分演变（见图 2-24）

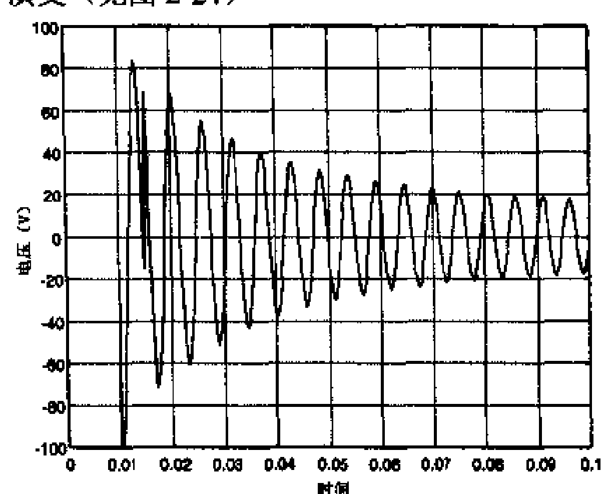


图 2-24 位置误差微分 $\mathrm{d}e(t)$

◆ 电磁控制电压（见图 2-25）

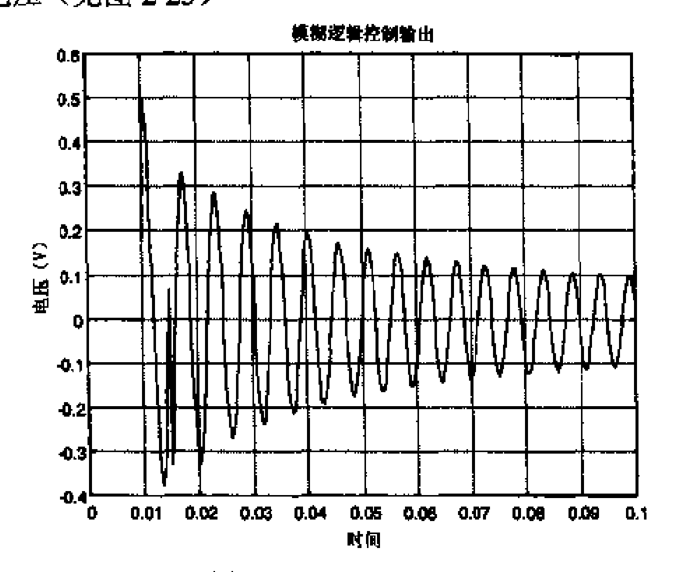
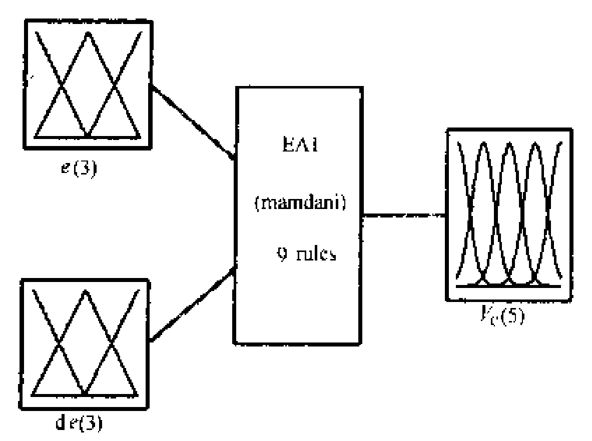


图 2-25 电磁控制电压

为在 0 附近增加过程阻尼而使用高斯隶属规则来修改输出变量的模糊设置。获得的新的修正结果保存在名为 EA1.fis 的文件中。



系统 EA1:2 输入,1 输出,9 法则

图 2-26 块位置控制系统

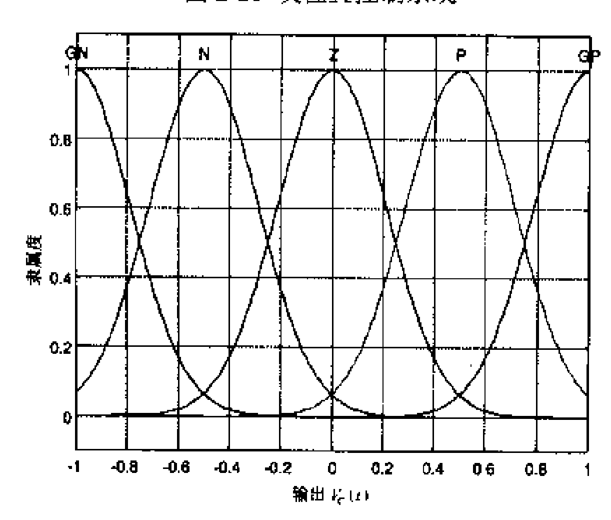


图 2-27 $V_C(t)$输出隶属规则

以前指令的过程响应有更大衰减。然而，位置误差仍是保证物体提升的平均值。物体的这个控制在 $e(t)$误差图上很清楚地表现出来。

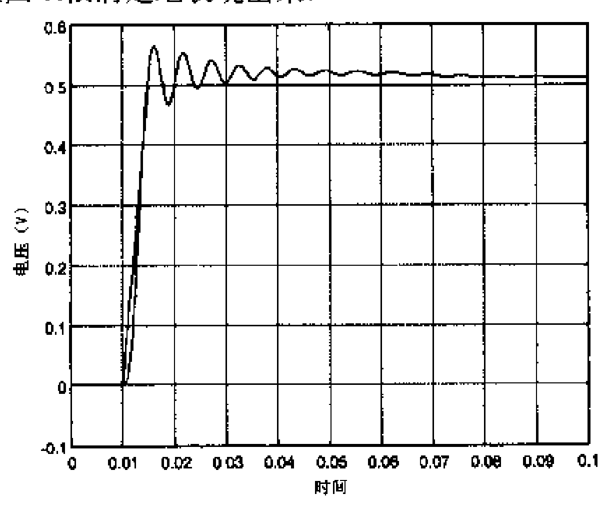


图 2-28 V_{π} 和位置 V_{λ}

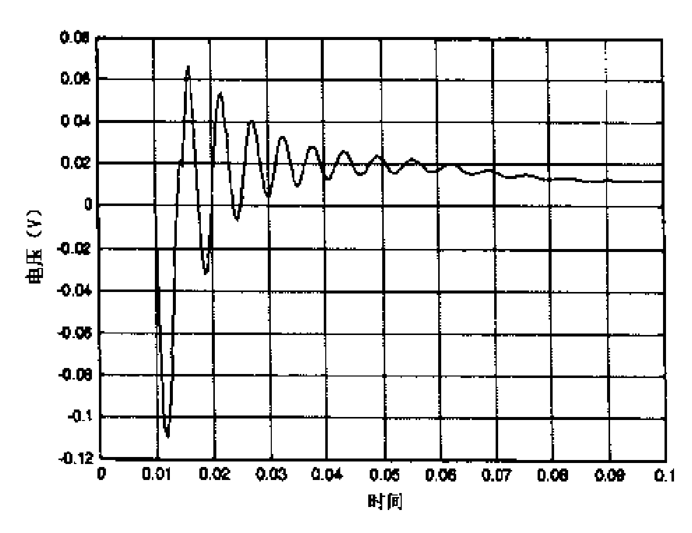


图 2-29　位置误差 $e(t)$

通过修改过程参数来检查修正系统的稳定性，如悬浮块的质量。将 M 从 1 kg 加到 2 kg，于是 K 和 τ 参数的电磁电流传递公式如下：

$$\begin{cases} b_0 = \dfrac{k_1}{M} = 7 \\ a_0 = -\dfrac{k_2}{M} = 1970 \end{cases}$$

在 SIMULINK 模型中也加上控制电流 I 的测量。

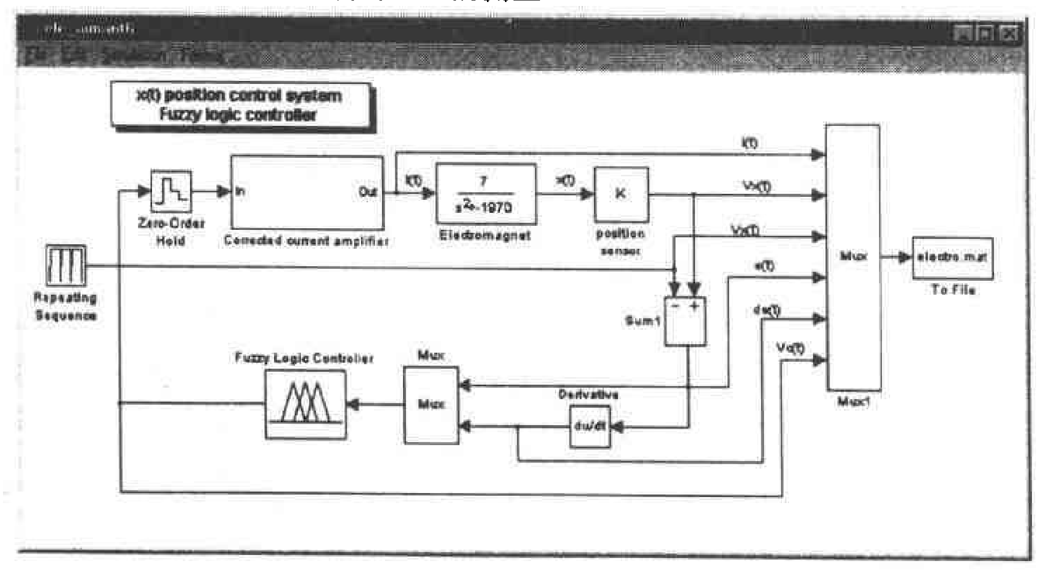


图 2-30　位置控制系统 $x(t)$ 模糊逻辑修正

这个模拟仍用相同的指令，但用了 EA.fis 第一调节器，导致增加了瞬态超调和很好的阻尼。这两个现象是由增加物块质量所引起的，本例质量是原来的 2 倍。

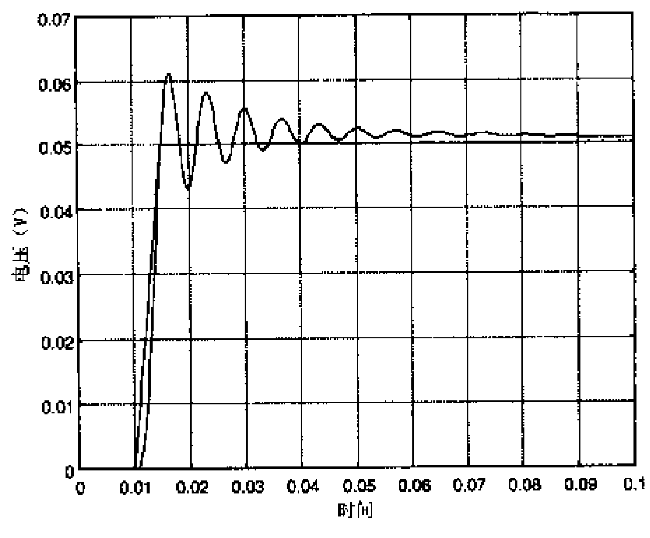


图 2-31 V_{xr} 和位置 V_x

图 2-32 位置误差 $e(t)$

悬浮块质量的增加使瞬间的误差振幅加大。

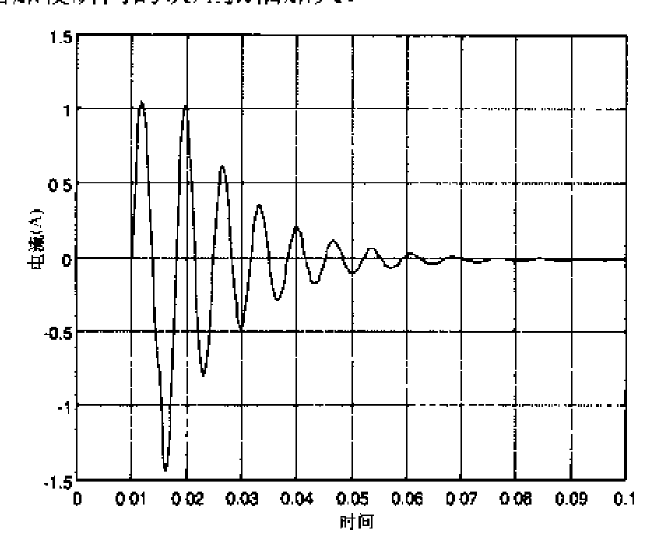


图 2-33 控制电流波动图

控制系统保持正常，这证明了模糊控制的稳定性。

应用 3　具有反转摆的小车

此应用的最终目的是调节小车位置，以使反转摆处于垂直位置。首先，考虑在不调整小车的 x 线性位置下让摆臂处于垂直位置。然后，将 x 和 θ 两因素考虑在内进行整体调节（见图 3-1）。

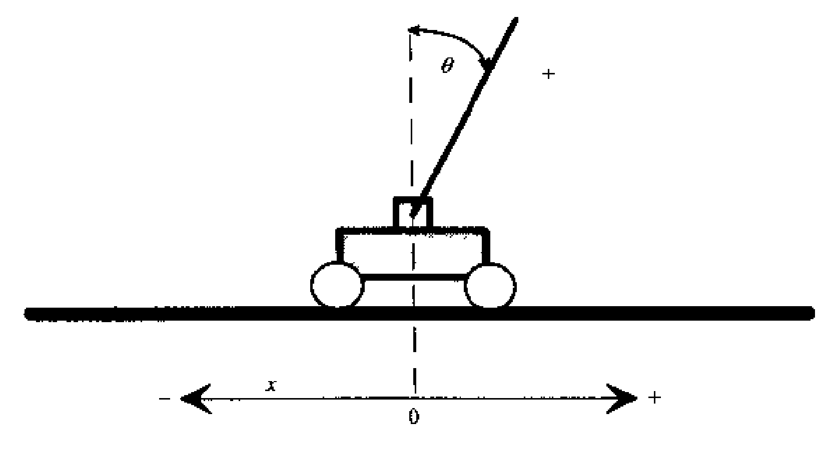


图 3-1　小车位置图

- 小车质量为 M，悬挂条质量为 m；
- 均匀断面悬挂摆长度为 1；
- 小车在 x 轴上移动，悬挂摆与它和小车连接点垂直方向呈 θ 角；
- 加在小车 x 轴上的力记为 $F(t)$；
- 假定系统无摩擦。

3.1　具有 2 个自由度的系统模型

用拉格朗日法定义如下方程：

$$\begin{cases} L = E_c - E_p \\ \dfrac{d}{dt}\left(\dfrac{\partial L}{\partial q'}\right) - \dfrac{\partial L}{\partial q} + \dfrac{\partial D}{\partial q'} = F_q \end{cases}$$

式中：

- q　$x(t)$和$\theta(t)$的自由度；
- D　由于摩擦而消失的能；
- F_q　θ ° 自由方向下的作用力；
- E_c　动能；
- E_p　势能。

3.1.1 移动时的系统动能

整个系统（车+摆）的动能用如下方程表示：

$$E_c = \frac{1}{2}mv^2 + \frac{1}{2}M\dot{x}^2 + \frac{1}{2}J\dot{\theta}^2$$

式中 v 代表摆重心的速度矢量。重心调节为：

$$x + \frac{l}{2}\sin\theta \quad （X\text{轴方向}）$$

$$\frac{l}{2}\cos\theta \quad （Y\text{轴方向}）$$

速度表达式定义为：

$$v_x = \dot{x} + \frac{l}{2}\dot{\theta}\cos\theta \quad （X\text{轴方向}）$$

$$v_y = -\frac{l}{2}\dot{\theta}\sin\theta \quad （Y\text{轴方向}）$$

于是

$$v^2 = {v_x}^2 + {v_y}^2 = \dot{x}^2 + \frac{l^2\dot{\theta}^2}{4} + \dot{x}l\dot{\theta}\cos\theta$$

动能表达式为：

$$E_c = \frac{1}{2}m\left(\dot{x}^2 + \frac{l^2\theta^2}{4} + \dot{x}l\dot{\theta}\cos\theta\right) + \frac{1}{2}\dot{M}x^2 + \frac{1}{2}J\dot{\theta}^2$$

3.1.2 系统势能

悬挂摆重心的势能为：

$$E_p = \frac{mgl}{2}\cos\theta$$

3.1.3 根据自由度 $q(t) = \theta(t)$ 的拉格朗日方程

系统拉格朗日 L 是动能和势能之差：

$$L = \frac{1}{2}m\left(\dot{x}^2 + \frac{l^2\dot{\theta}^2}{4} + \dot{x}l\dot{\theta}\cos\theta\right) + \frac{1}{2}M\dot{x}^2 + \frac{1}{2}J\dot{\theta}^2 - \frac{mgl}{2}\cos\theta$$

上式对 $\dot{\theta}$ 的偏微分为

$$\frac{\partial L}{\partial \dot{\theta}} = \frac{ml^2}{4}\dot{\theta} + J\dot{\theta} + \frac{ml}{2}\dot{x}\cos\theta$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = \frac{ml^2}{4}\ddot{\theta} + J\ddot{\theta} + \frac{ml}{2}\ddot{x}\cos\theta - \frac{ml}{2}\dot{x}\dot{\theta}\sin\theta$$

$$\frac{\partial L}{\partial \theta} = -\frac{ml}{2}\dot{x}\dot{\theta}\sin\theta + \frac{mgl}{2}\sin\theta$$

获得如下第一拉格朗日方程

$$\left(\frac{ml^2}{4}+J\right)\ddot{\theta}+\frac{ml}{2}(\ddot{x}\cos\theta-g\sin\theta)=0$$

3.1.4 根据自由度 $q(t)=x(t)$ 的拉格朗日方程

对 $\dot{x}$ 的拉格朗日偏微分为

$$\frac{\partial L}{\partial \dot{x}}=(M+m)\dot{x}+\frac{ml}{2}\dot{\theta}\cos\theta$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right)=(M+m)\ddot{x}+\frac{ml}{2}\ddot{\theta}\cos\theta-\frac{ml}{2}\dot{\theta}^2\sin\theta$$

$$\frac{\partial L}{\partial x}=0$$

于是，第二拉格朗日方程为

$$(M+m)\ddot{x}+\frac{ml}{2}(\ddot{\theta}\cos\theta-\dot{\theta}^2\sin\theta)=F(t)$$

3.1.5 操作点附近的线性模型

如果只考虑θ在操作点$\theta_0=0$附近的细微变化，根据悬挂摆的垂直位置，记

$$\begin{cases}\cos\theta\approx 1\\ \sin\theta\approx\theta\end{cases}$$

还可简化如下公式

$$\ddot{\theta}\cos\theta-\dot{\theta}^2\sin\theta$$

为

$$\frac{\mathrm{d}}{\mathrm{d}\theta}(\dot{\theta}\cos\theta)\approx\frac{\mathrm{d}}{\mathrm{d}\theta}(\dot{\theta}\cos\theta)\approx\ddot{\theta}$$

接着得到线性系统

$$\begin{cases}(M+m)\ddot{x}+\dfrac{ml}{2}\ddot{\theta}=F(t)\\ \left(J+\dfrac{ml^2}{4}\right)\ddot{\theta}+\dfrac{ml}{2}(\ddot{x}-g\theta)=0\end{cases}$$

记悬挂摆的惯性 $J=\dfrac{ml^2}{12}$，得到

$$\begin{cases}(M+m)\ddot{x}+\dfrac{ml}{2}\ddot{\theta}=F(t)\\ \dfrac{ml^2}{3}\ddot{\theta}+\dfrac{ml}{2}(\ddot{x}-g\theta)=0\end{cases}$$

3.2 线性过程状态模型

从前述系统的第一方程，得到

$$\ddot{\theta}(t)=\frac{2}{ml}\left[F(t)-(M+m)\ddot{x}\right]$$

在得到第二方程后，有

$$\frac{2l}{3}\left[F(t)-(M+m)\ddot{x}\right]+\frac{ml}{2}(\ddot{x}-g\theta)=0$$

于是

$$\ddot{x}(t)=\frac{4}{m+4M}F(t)-\frac{3gm}{m+4M}\theta(t)$$

$$\ddot{\theta}(t)=\frac{2}{ml}F(t)-\frac{2(M+m)}{ml}\left[\frac{4}{m+4M}F(t)-\frac{3gm}{m+4M}\theta(t)\right]$$

$$\ddot{\theta}(t)=\frac{-6}{l(m+4M)}F(t)+\frac{6g(m+M)}{l(m+4M)}\theta(t)$$

系统用 2 个 2 级微分方程概括，系统状态矢量如下：

$$X=[\theta \quad \dot{\theta} \quad x \quad \dot{x}]$$

系统模型用如下方程描述：

$$\begin{cases}\dot{X}=AX+Bf\\ y=CX\end{cases}$$

$$\begin{bmatrix}\dot{\theta}\\ \ddot{\theta}\\ \dot{x}\\ \ddot{x}\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & 0\\ \frac{6g(m+M)}{l(m+4M)} & 0 & 0 & 0\\ 0 & 0 & 0 & 1\\ \frac{-3gm}{m+4M} & 0 & 0 & 0\end{bmatrix}\times\begin{bmatrix}\theta\\ \dot{\theta}\\ x\\ \dot{x}\end{bmatrix}+\begin{bmatrix}0\\ \frac{-6}{l(m+4M)}\\ 0\\ \frac{4}{m+4M}\end{bmatrix}\times f(t)$$

<u>数据</u>

M=2 kg　　m=0.1 kg　　l=0.5 m　　g≈10 m/s^2

$$\boldsymbol{A}=\begin{bmatrix}0 & 1 & 0 & 0\\ 31.1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1\\ -0.37 & 0 & 0 & 0\end{bmatrix}\qquad \boldsymbol{B}=\begin{bmatrix}0\\ -1.48\\ 0\\ 0.494\end{bmatrix}$$

首先调整悬挂摆的角度在垂直位置且独立于位置 x。为此，采用模糊逻辑控制算法。这是一个必须得到系统离散模型的离散控制。

3.3 离散模型的版本与检测

下面的文件 endul_invl.m 中由 c2dt 函数计算出状态离散模型。考虑到过程的动态特性，取采样周期为 0.01 s。

pendul_inv1.m

```
% Inverted pendulum
% Modelling in discrete state space

% system parameters
```

```
M = 1; m = 0.1;
l = 0.5; g = 10;
k1 = 6*(M+m)*g/(l*(4*M+m));
k2 = -3*m*g/(4*M+m);
k3 = -6/(l*(4*M+m));
k4 = 4/(4*M+m);

% Sampling Period
Te = 0.01;

% Continuous state representation matrices
A = [0 1 0 0 ;k1 0 0 0 ; 0 0 0 1 ; k2 0 0 0 ];
B = [0; k3; 0; k4];
C = [1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];

printmat(A,'A matrix')
printmat(B,'B matrix ')
printmat(C,'C matrix ')
```

```
A matrix =
          ----1----         ----2----       ----3----       ----4----
--1-->        0             1.00000             0               0
--2-->    31.11111              0               0               0
--3-- >       0                 0               0           1.00000
--4-->    -0.37037              0               0               0

B matrix =
                    ----1-----
          --1-->         0
          --2-->   -1.48148
          --3-- >        0
          --4-->    0.49383

C matrix =
              ----1----       ----2----       ----3----       ----4----
--1-->        1.00000             0               0               0
--2-->            0           1.00000             0               0
--3-- >           0               0           1.00000             0
--4-->            0               0               0           1.00000

% discrete state space representation
[Ad,Bd,Cd,Dd] = c2dt(A,B,C,Te,0);
printmat(Ad,'Ad matrix')
printmat(Bd,'Bd matrix')
printmat(Cd,'Cd matrix')
```

```
Ad matrix =
        ----1----      ----2----      ----3----      ----4----
--1-->   1.00156        0.01001           0              0
--2-->   0.31127        1.00156           0              0
```

```
--3--> -1.85233e-0.05  -6.17380e-008  1.00000   0.01000
--4--> -0.00371        -1.85233e-005  0         1.00000

Bd matrix =
                    ----1-----
         --1-->     -7.40933e-005
         --2-->     -0.01482
         --3--  >    2.46916e-005
         --4-->      0.00494

Cd matrix =
         ----1----     ----2----    ----3----    ----4----
--1-->   1.00000       0            0            0
--2-->   0             1.00000      0            0
--3--  > 0             0            1.00000      0
--4-->   0             0            0            1.00000
```

◆ 反转摆的 SIMULINK 模型（见图 3-2）

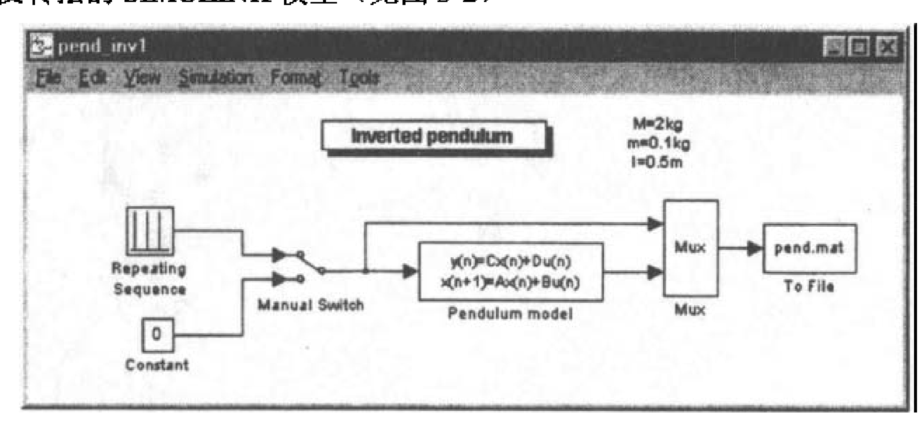


图 3-2 反转摆的 SIMULINK 模型

下面的文件 pendu1_inv2.m 呈现的是对 0.1s 持续时间和幅度为 0.1N 的作用力脉冲 $f(t)$ 的响应。

pendul_inv2.m file

```
% Inverted pendulum
% Test of the discrete model in open loop
% signals recuperation
load pend.mat
t = signals(1,:);
f = signals(2,:);
teta = signals(3,:);
dteta = signals(4,:);
x = signals(5,:);
dx = signals(6,:);

% drawing control and teta(t) response signals
figure(1)
hf = line(t,f(:));
xlabel('Time')
ylabel('Force in N')
```

```
axis([0 1 0 0.12])
axet = axes('Position',get(gca,'Position'),...
          'XAxisLocation','bottom',...
          'YAxisLocation','right','Color','none',...
          'XColor','k','YColor','k');
ht = line(t,teta,'color','r','parent',axet);
ylabel('Angle evolution')
title('Response \Theta(t) in rd to a f(t) step of 0.1 N')
gtext('f(t)\rightarrow'), gtext('\leftarrow \theta(t)')

% drawing control and teta(t) response signals
figure(2)
hf = line(t,f(:));
xlabel('Time')
ylabel('Force in N')
axis([0 1 0 0.12])
axet = axes('Position',get(gca,'Position'),...
          'XAxisLocation','bottom',...
          'YAxisLocation','right','Color','none',...
          'XColor','k','YColor','k');
ht = line(t,teta,'color','r','parent',axet);
ylabel('Evolution of the derivative of the angle')
title('Response \Theta(t) in rd to a f(t) step of 0.1 N')
gtext('f(t)\rightarrow'), gtext('\leftarrow \theta''(t)')

% drawing control and x response signals

figure(3)
hf = line(t,f(:));
xlabel('Time')
ylabel('Force in N')
axis([0 1 0 0.12])
axet = axes('Position',get(gca,'Position'),...
          'XAxisLocation','bottom',...
          'YAxisLocation','right','Color','none',...
          'XColor','k','YColor','k');
ht = line(t,x,'color','r','parent',axet);
ylabel('Evolution of the x position')
title('x Response in meters to a f(t) step of 0.1 N')
gtext('f(t)\rightarrow'), gtext('\leftarrow x(t)')

% drawing control and x' response signals
figure(4)
hf = line(t,f(:));
xlabel('Time')
ylabel('Force in N')
axis([0 1 0 0.12])
axet = axes('Position',get(gca,'Position'),...
          'XAxisLocation','bottom',...
          'YAxisLocation','right','Color','none',...
          'XColor','k','YColor','k');
```

```
ht = line(t,dx,'color','r','parent',axet);
ylabel('Evolution of the derivative of the x(t) position')
title('x' Response to a f(t) step of 0.1 N')
gtext('f(t)\rightarrow')
gtext('\leftarrow x''(t)')
```

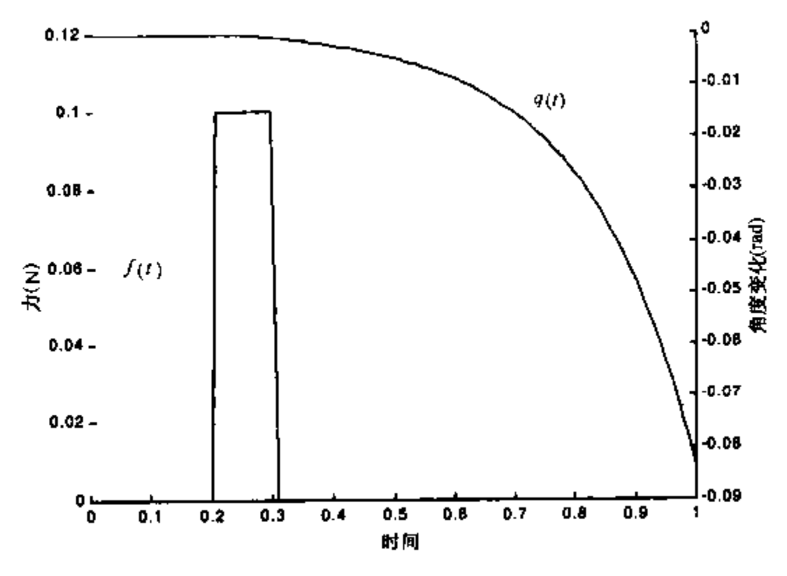


图 3-3　$f(t)$下的响应 $Q(t)$

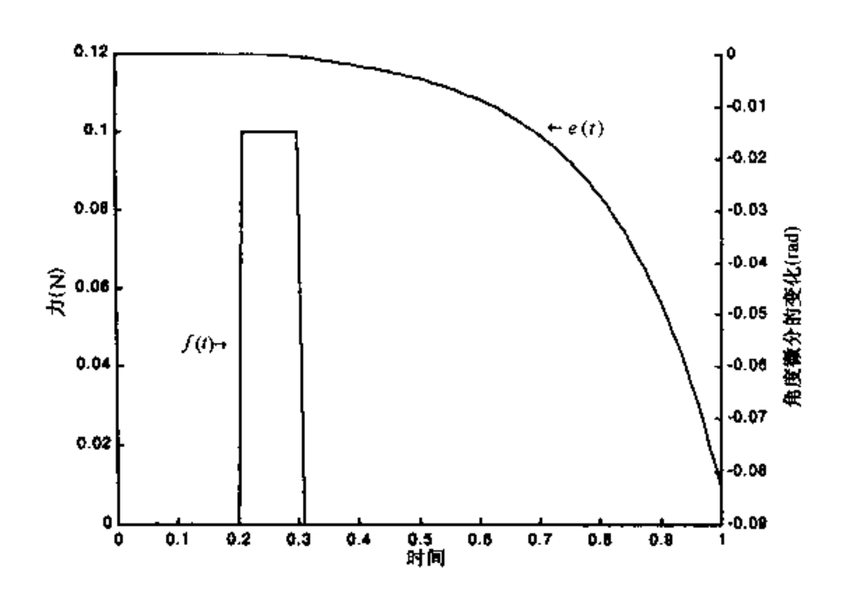


图 3-4　$f(t)$下的响应 $\theta(t)$

对于正力作用，摆以负 $\theta(t)$ 角摆动，速度增加。

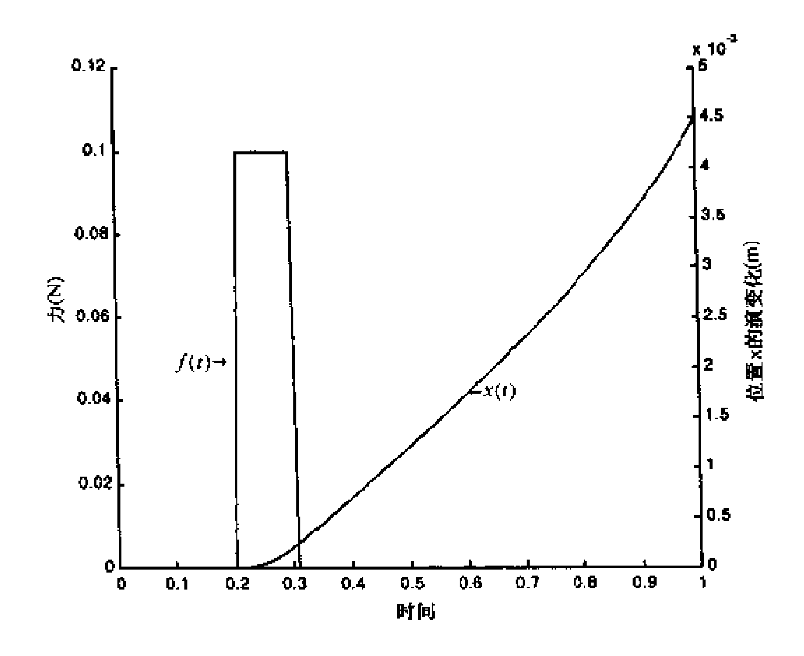


图 3-5 $f(t)$下的响应 x

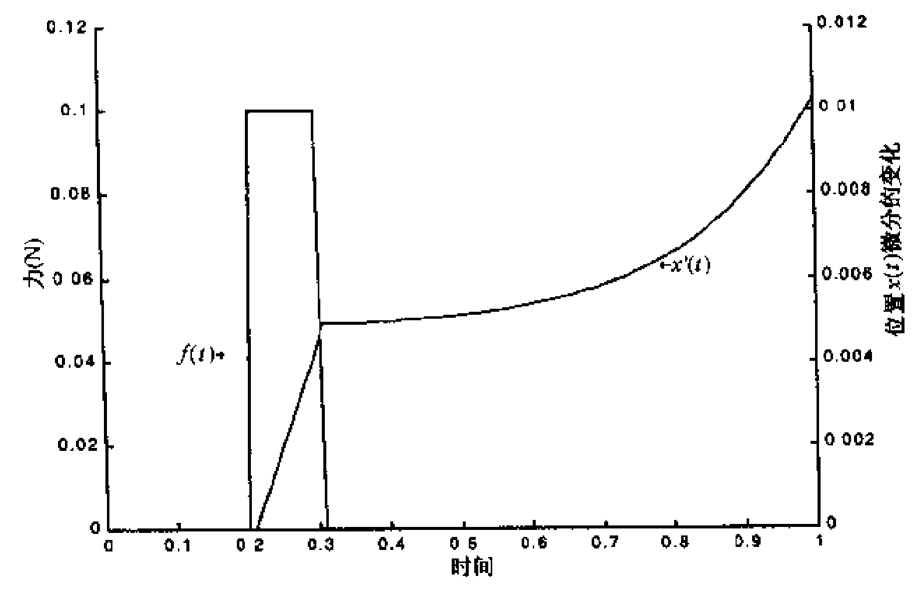


图 3-6 $f(t)$下的响应 x'

在常力作用下，位置呈抛物线变化。在此位置时取消作用，因为假定系统无摩擦，所以小车仍保持前进。还可检测在初始矢量情况下的响应过程。

注意 $\boldsymbol{X}_0=\begin{bmatrix}0.1\\0\\0\\0\end{bmatrix}$，相当于除车和摆的速度为零外，摆的初始倾斜度为 0.1rad，小车的位置在 x=0 处。

文件 pendul_inv2_1.m 可呈现出给摆模型加上初始矢量、零设定值上不平衡手动开关这一情况下的输出状况。

pendul_inv2-1.m

```
% Inverted pendulum
% Test of discrete model in open loop

% signals recuperation
load pend.mat
t = signals(1,:);
f = signals(2,:);
teta = signals(3,:);
dteta = signals(4,:);
x = signals(5,:);
dx = signals(6,:);
% drawing the teta(t) and teta'(t) responses
figure(1)
title1 = '\theta(t) and \theta''(t) responses';
title2 = 'to initial conditions' ;
title(strcat(title1, title2))
plot(t,teta)
hold on
plot(t,dteta,'r')
hold off
xlabel('Time')
title('\Theta(t) and \Theta''(t) responses to initial conditions')
gtext('\theta(t)\rightarrow'),gtext('\leftarrow \theta''(t)')
gtext(' initial \theta = 0.1 rd')

% drawing x(t) and x'(t) responses
figure(2)
plot(t,x), hold on
plot(t,dx,'r'), hold off
xlabel('Time')
title('x(t) and x''(t) responses to initial conditions')
gtext('x(t)\rightarrow')
gtext('\leftarrow x''(t)')
gtext('initial \theta = 0.1 rd')
```

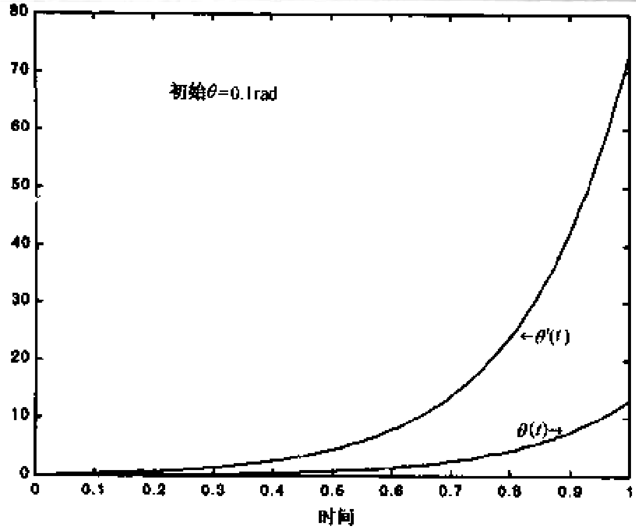


图 3-7 初始条件下的$\theta(t)$和$\theta'(t)$响应

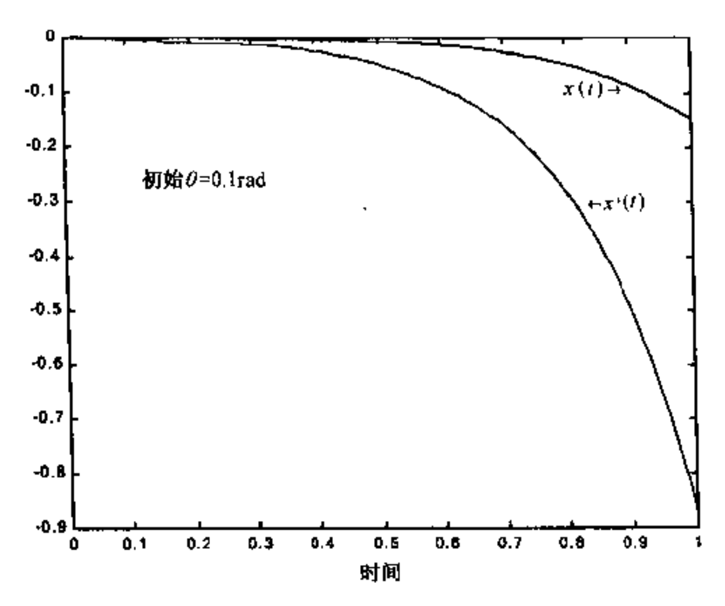


图 3-8　初始条件下的 $x(t)$和 $x'(t)$响应

在力 $f(t)$为零、初始相位θ=0.1rad 情况下，拖动摆向下，使车向左移动。有必要记住线性模型只有在θ很小的情况下才有用。

3.4　角位置$\theta(t)$的模糊调整

为使摆不取决于小车的位置 x 而处于垂直位置，可在零设定点调节角位置$\theta(t)$。因此，模糊控制器呈现 2 个输入θ和$\dot{\theta}$，以及反馈力输出 $f_r(t)$。相应的 SIMULINK 模型如图 3-9 所示。

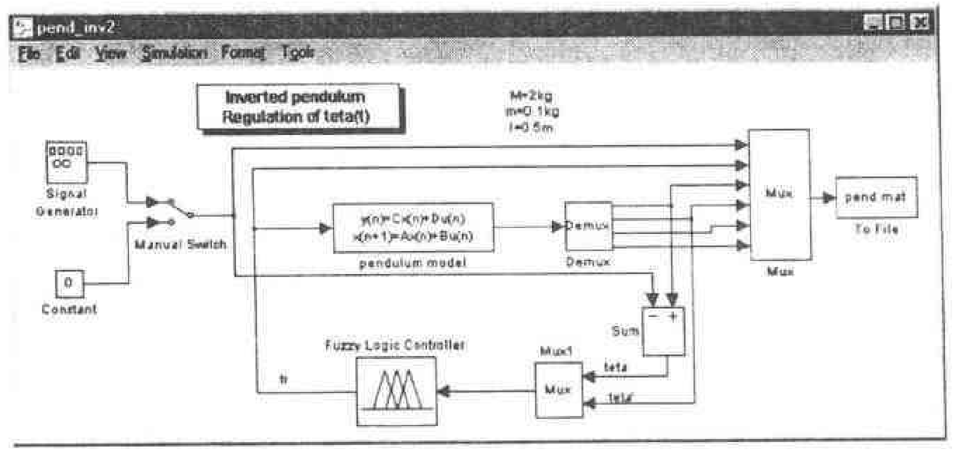


图 3-9　SIMULINK 模型

作用“模糊逻辑工具箱”的绘图编辑器，可以定义模糊控制器。这由 Ruleview 命令来打开。

模糊控制定义的 3 个不同阶段分别如下：

- 输入模糊化，定义隶属函数；
- 定义推理规则，由从模糊输入变量中定义模糊输出变量组成，使用算子和蕴含法则；

- 输入非模糊化，使我们能够从定义的模糊输出集合中找到输出信号估计。

3.4.1 输入模糊化，隶属函数定义

这里有 2 个输入：θ 和它的变分 $\dot{\theta}$。每个输入使用 2 个规则，因此有 4 个规则。对于模糊化操作，一些可预知的函数将第一次被使用，如 Gbellmf 函数或广义 Bell 曲线。使用绘图编辑器完成的模糊控制在文件 fuz_teta 下保存。

选择值域：

$$-0.3 < \theta < 0.3$$

$$-1 < \dot{\theta} < 1$$

为保证摆在微小的角度附近，所以它不能摆得过度。文件 fuzzy_teta.m 中的下列命令可以检验在图中定义的隶属函数。

fuzzy_teta.m file

```
% fuzzy matrix loading
fismat1 = readfis('fuz_teta')
% membership functions reading
figure(1)
plotmf(fismat1,'input',1), grid
title('\theta(t) input membership functions')
xlabel('\theta(t) input')
ylabel('membership degree')
figure(2)
plotmf(fismat1,'input',2), grid
title('\theta''(t) membership functions')
xlabel('\theta(t)'' input')
ylabel('membership degree')
```

```
>> fuzzy_teta
        name:   `fuz_teta`
        type:   `mamdain`
   andmehtod:   `min`
    ormethod:   `max`
defuzzMethod:   `centroid`
    imMethod:   `min`
   aggMethod:   `max`
      input:  [1x2 struct]
     output:  [1x1 struct]
       rule:  [1x4 struct]
```

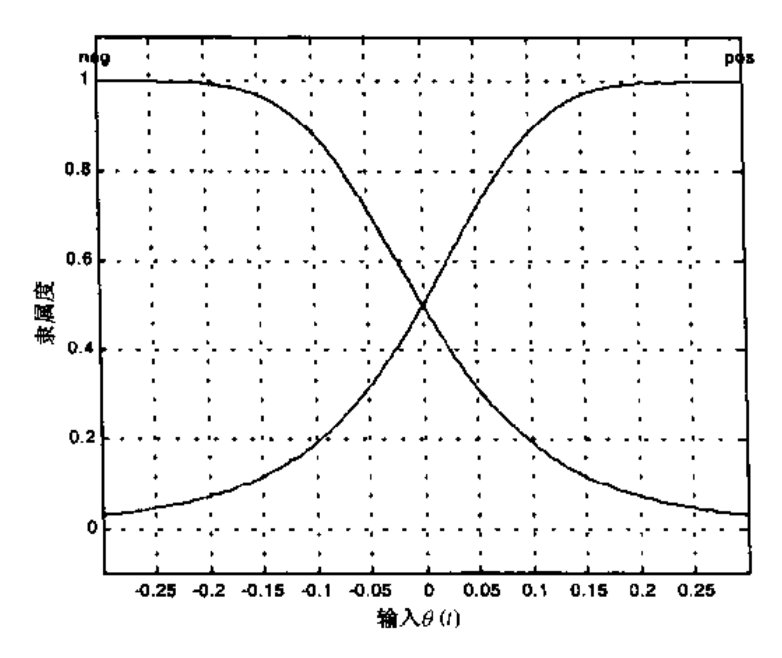


图 3-10　输入隶属函数$\theta(t)$

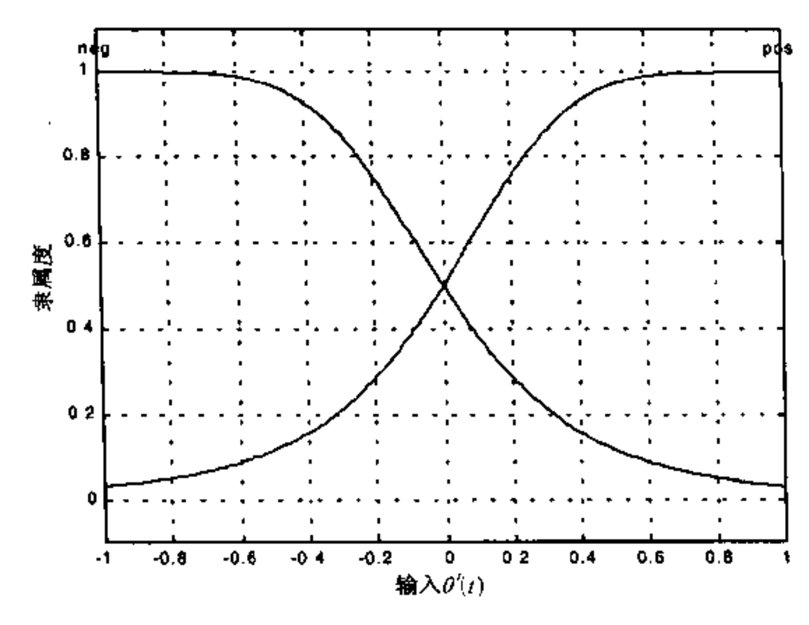


图 3-11　输入隶属函数$\theta'(t)$

因输入的 neg 规则，下列命令可定义输出的隶属函数。

```
x = (-0.3 : 0.03 : 3)';
neg = gbellmf(x,[0.3  2.5  0.3])
```

这里使用很常用的 min-max 方法。它的使用和操作很简单，一些简单命令（比较和假设）只需要很短的计算时间，特别是在使用微控制执行的情况下。同样的原因，我们使用典型的 min 蕴含。对 4 个输出隶属函数用 trimf 三角隶属函数。下面的命令使在绘图编辑器中定义的隶属函数可视化。

```
% output membership functions drawing
figure(3)
plotmf(fismat1,'output',1), grid
title('fr(t) membership functions')
```

```
xlabel('fr(t) output in N')
ylabel('membership degree')
```

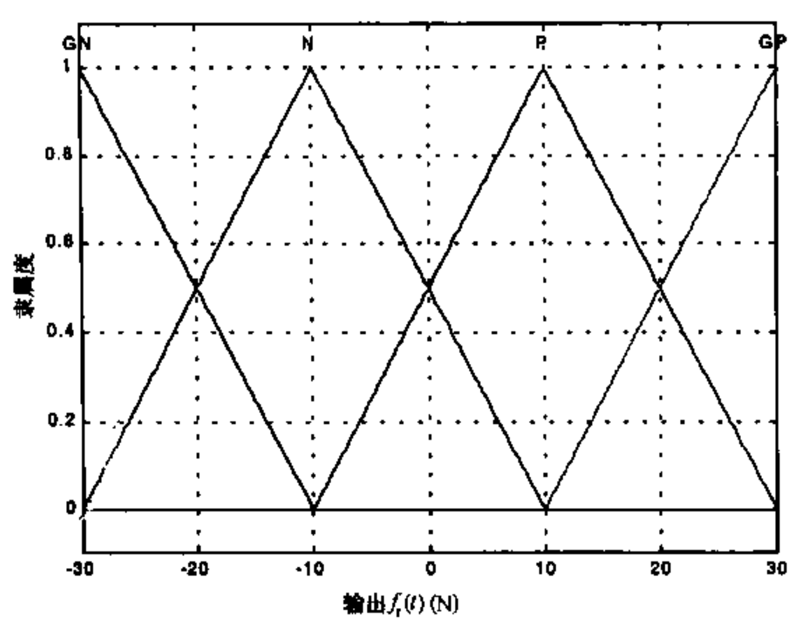


图 3-12　隶属函数$f_r(t)$

这相当于下面 GN 规则命令的输出。

```
x = (-30:1:30)
GN=trimf(X,[-50 -30 -10])
```

3.4.2　推理规则定义，非模糊化

限制每个输入用 2 个规则，只能使用 4 个推理规则。为建立这些规则，使用 2 种方法：Mamdani 规则和 Sugeno 规则。第 1 种情况是，规则数不重要时使用 Mamdani 方法。当处理规则数变得重要时，最好使用 Sugeno 规则。系统可用如下 4 种情况表示在达到垂直位置附近的摆的运动。

输 入 状 态	在θ=0 附近的摆等阶运动
$\theta>0$，$\theta'>0$	在正θ 边离开
$\theta<0$，$\theta'<0$	在负θ 边离开
$\theta>0$，$\theta'>0$	靠近正θ 边
$\theta<0$，$\theta'<0$	靠近负θ 边

这 4 种情况可得到显示与隶属函数相联系的规则的如下图形。

力f (θ / $\dot{\theta}$)	负	正
负	GN	P
正	N	GP

由绘图编辑器，这 4 个规则总是被定义（见图 3-13）。

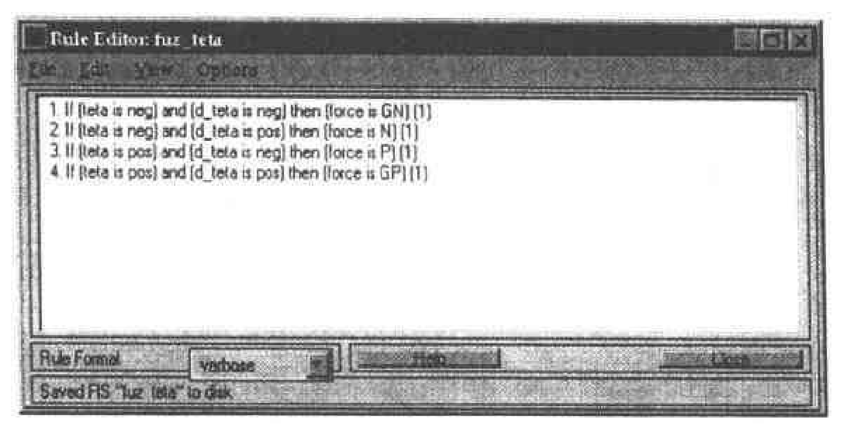


图 3-13 用绘图编辑器编辑规则

使用 showrule 命令也可以检验。

◆ 动词方式

```
%editing the inference rules
showrule(fismat,1:4,'verbose')
1.If  (teta is neg)  and  (d_teta is neg)  then  (force is GN)  (1)
2.If  (teta is neg)  and  (d_teta is pos)  then  (force is N)   (1)
3.If  (teta is pos)  and  (d_teta is neg)  then  (force is P)   (1)
4.If  (teta is pos)  and  (d_teta is pos)  then  (force is GP)  (1)
```

◆ 索引方式

```
%editing the inference rules
showrule(fismat, 1 : 4, `indexed`)
1 1, 1 (1) : 1
1 2, 2 (1) : 1
2 1, 3 (1) : 1
2 2, 4 (1) : 1
```

◆ 符号方式

```
%editing the inference rules
showrule(fismat, 1 : 4, `indexed`)
1.(teta= =neg)&(d_teta= =neg)=>(force=GN)  (1)
2. (teta==neg)&(d_teta= =pos)=>(force=N)  (1)
3. (teta= =pos)&(d_teta= =neg)=>(force=P)  (1)
4. (teta= =pos)&(d_teta= =pos)=>(force=GP)  (1)
```

最后，只需要比较-假设，使用的集合方法就成为最大的一个。

对于非模糊化，使用重心方法。接着借助于表面可视函数或规则证明可视模糊控制的操作。

在绘图编辑器里或通过 gensurf 函数可获得表面图。

```
gensurf(fismat)
title('Surface graphic of the angle controller ')
```

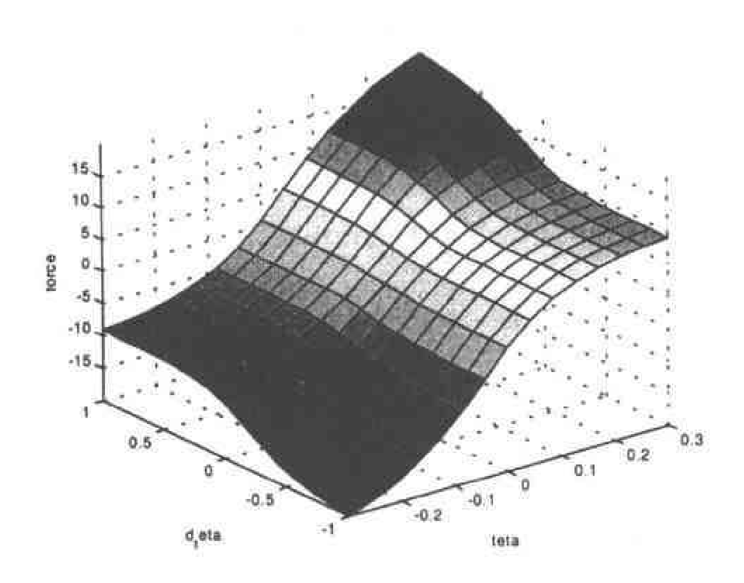


图 3-14　角控制器的表面图

可以看到，当 2 个输入为零时输出取消，这相当于摆处于垂直位置。在输入变化时，输出并没引起很强的非连续，这象征了控制的适应性。同时可以看到，2 个输入与对应于 2 种情况下摆处于远离位置同样的信号一样时，力是很重要的。

可以用绘图编辑器或 ruleview 函数来检测规则。

```
% Test of the fuzzy controller
ruleview(fismat1)
```

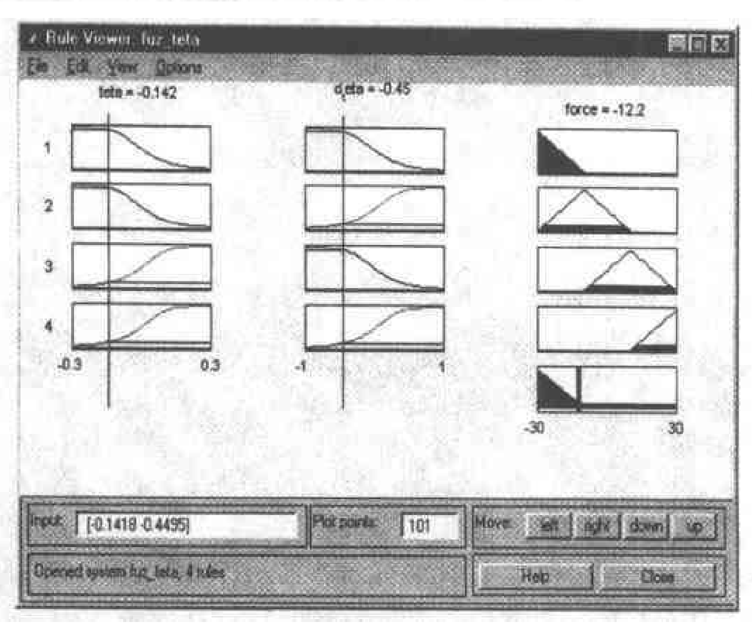


图 3-15　查看规则

可以看到，与每个规则对应的可视化线使用根据 min 方法的蕴含规则。同样，当从头到底的可视化输出栏时，集合方法正好是 max 结果。最后，重心非模糊化方法对应于 $\theta=-0.142\text{rad}$ 及它的变量 $\theta'=-0.45\text{rad/s}$ 时力为-12.2N 的情况。很容易证明替换输入索引时的控制器的工作。

3.4.3 获得模糊控制

可以在 SIMULINK 的“模糊控制”块中指定含有控制参数的矩阵名，在此，指定为模糊 fismat 矩阵。

文件 pendul_inv4.m 表示对角设置点θ_c的过程响应。第 1 次设置点为零，证明控制在初始状态：

- 0.1rad 的摆倾斜；
- 0.2rad/s 的摆速度。

$$X_0=\begin{bmatrix}0.1\\0.2\\0\\0\end{bmatrix}$$

pendul_inv4.m

```
% Inverted pendulum, regulation of teta
% signals recuperation
load pend.mat
t = signals(1,:); tetac = signals(2,:);
f = signals(3,:);
teta = signals(4,:); dteta = signals(5,:);
x = signals(6,:); dx = signals(7,:);

% angle drawing
figure(1)
hf = line(t,teta(:));
xlabel('Time'), ylabel('Angle in rd')
axet = axes('Position',get(gca,'Position'),...
   'XAxisLocation','bottom',...
   'YAxisLocation','right','Color','none',...
   'XColor','k','YColor','k');
ht = line(t,dteta,'color','r','parent',axet);
ylabel('Angle variation in rd/s')
title('Response to the set-point \theta = 0')
gtext('\theta(t)'), gtext('\theta''(t)')
gtext ('\initial\theta=0.1rd') , gtext ('\yntial\theta=0.2rd/s')

% position drawing
figure(2), hf = line(t,x(:));
xlabel('Time'), ylabel('Position in m')
axet = axes('Position',get(gca,'Position'),...
          'XAxisLocation','bottom',...
          'YAxisLocation','right','Color','none',...
          'XColor','k','YColor','k');
ht = line(t,dx,'color','r','parent',axet);
ylabel('position variation')
title('Response to the set-point \theta = 0')
gtext('x(t)'),gtext('x''(t)'),gtext (`initial \ theta = 0.1 rd`)
gtext (`initial \ theta`` = 0.2 rd/s`)
```

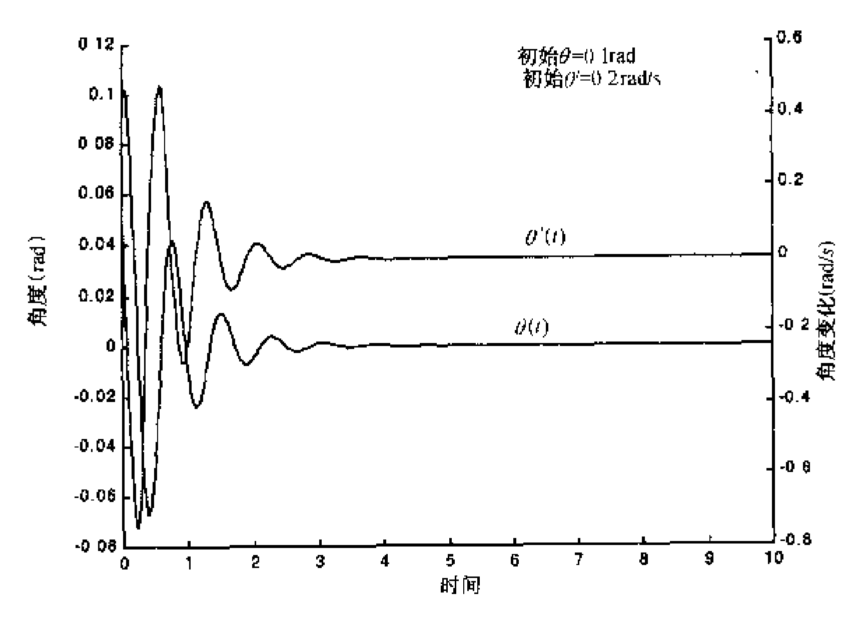


图 3-16　在设定点$\theta=0$时的角度变化响应

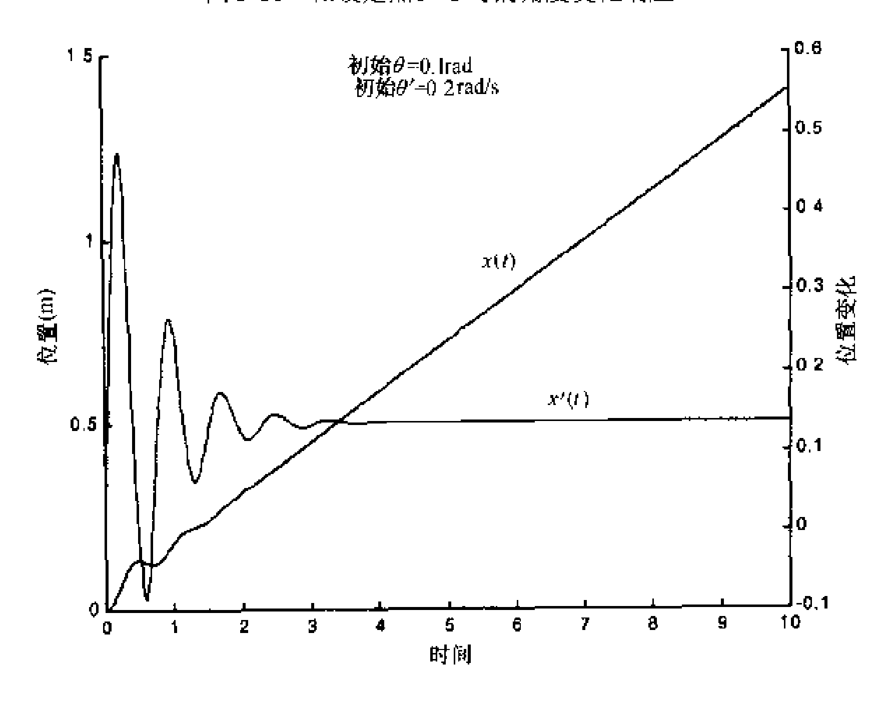


图 3-17　在设定点$\theta=0$的位置变化响应

初始状态在 0.1 rad 和 0.2 rad/s 速度下的摆在 2 s 后回到垂直位置，为此，车应向右移动大约 30 cm。可以看到，一旦摆在垂直位置，车甚至会在没有力 $f(t)$下继续移动，这都是在系统无能量损失下产生的现象。现在证明在达到设定点信号时加上幅度为 0.05 rad 的方波信号的过程响应。

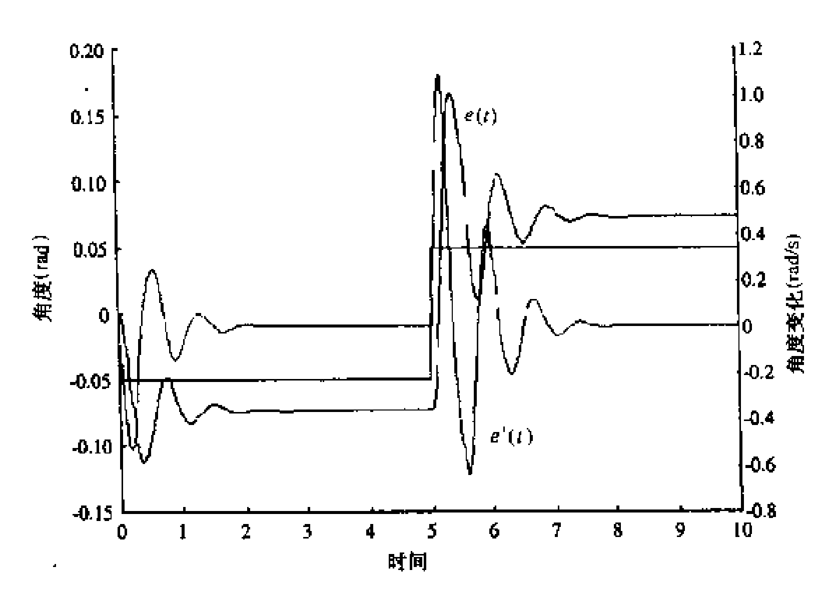


图 3-18 在设定点θ(t)时的角度变化响应

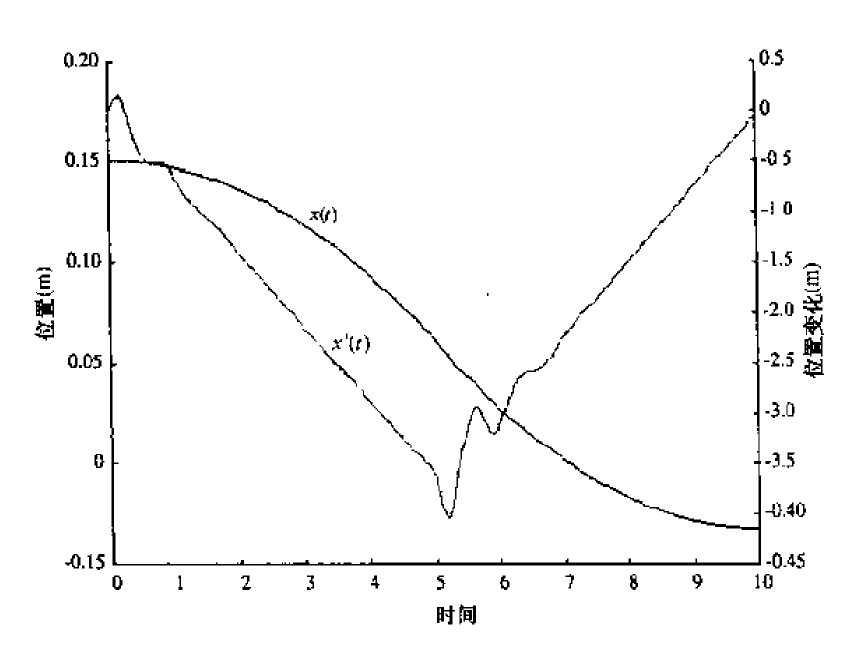


图 3-19 在设定点θ(t)的位置变化响应

系统在非零稳态误差下得到设定点信号。

现在仍以模糊控制的简化形式，并通过梯形模糊（作用于 2 输入的 trapmf 函数）来修改这一情况。

由绘图编辑器修改的模糊控制保存在文件 fuz_teta0.fis 下。

以下命令能够使修改的输入隶属函数可视化。

```
% loading the fuzzy matrix of the controller
fismat1 = readfis('fuz_teta0');
% membership functions drawing
figure(1)
plotmf(fismat1,'input',1), grid
title(`inputs \theta(t) membership functions')
xlabel('\theta(t) input')
```

```
ylabel('membership degree')
figure(2)
plotmf(fismat1,'input',2), grid
title ('inputs \theta(t) membership functions')
xlabel('\theta''(t) input')
ylabel('membership degree')
```

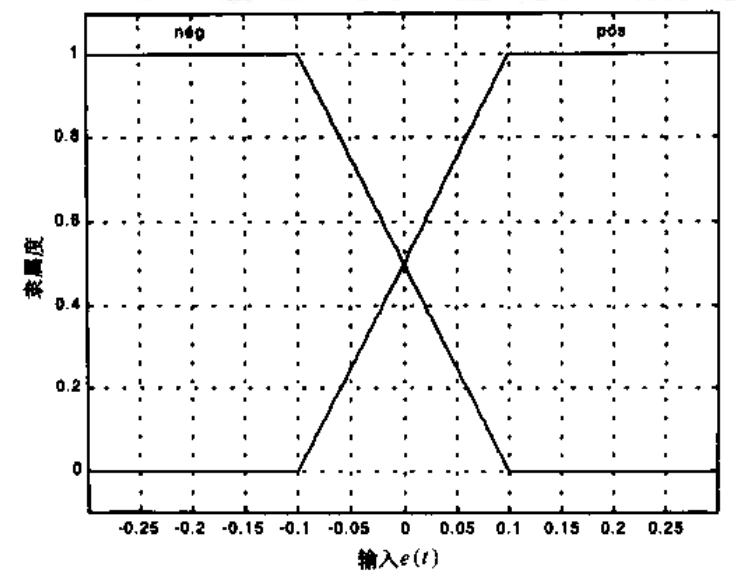


图 3-20 输入$\theta(t)$隶属函数

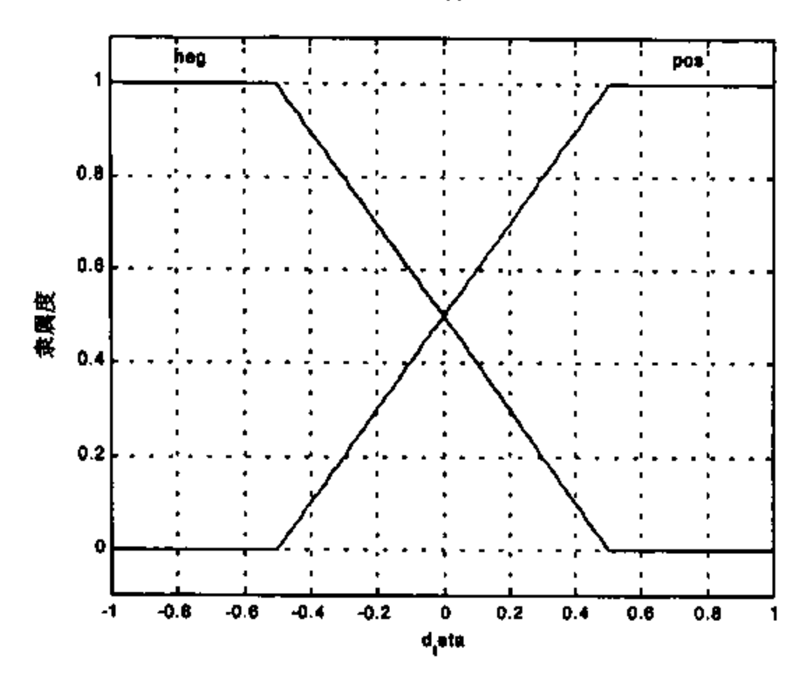


图 3-21 输入$\theta'(t)$隶属函数

得到如图 3-22 所示的图形表面。

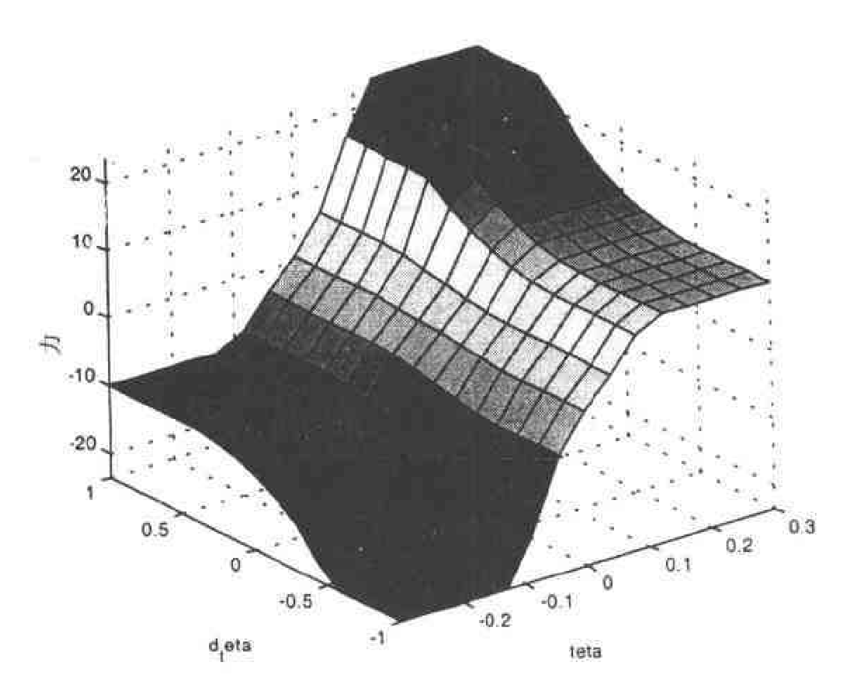


图 3-22 梯形隶属函数表面图

在输入聚集在设置点 $\theta_0=0$ 附近时输出很快达到饱和。过程跟踪响应显示在下面的结果图中。

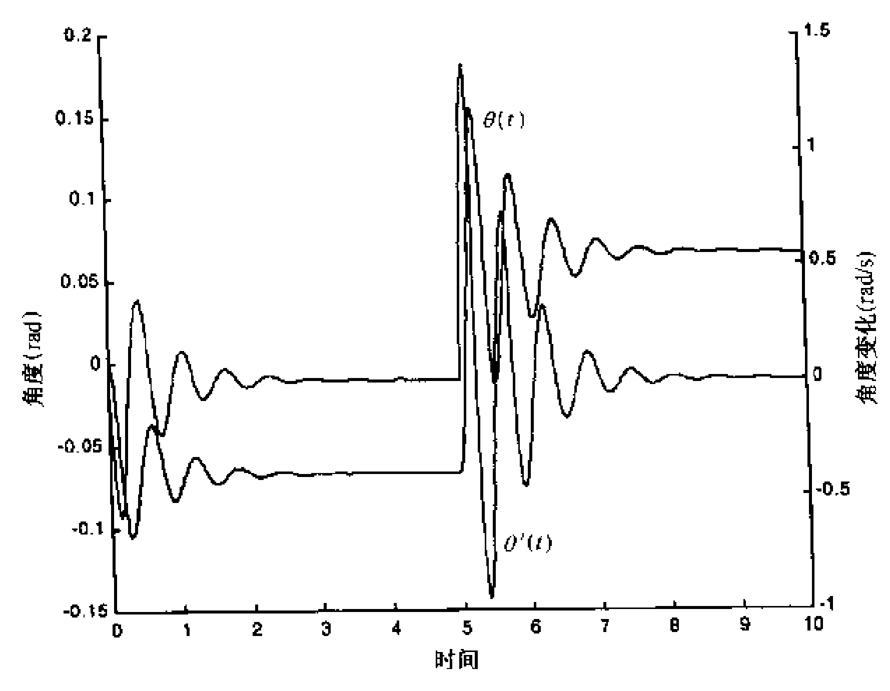


图 3-23 在设定点 $\theta=0$ 时的角度变化响应

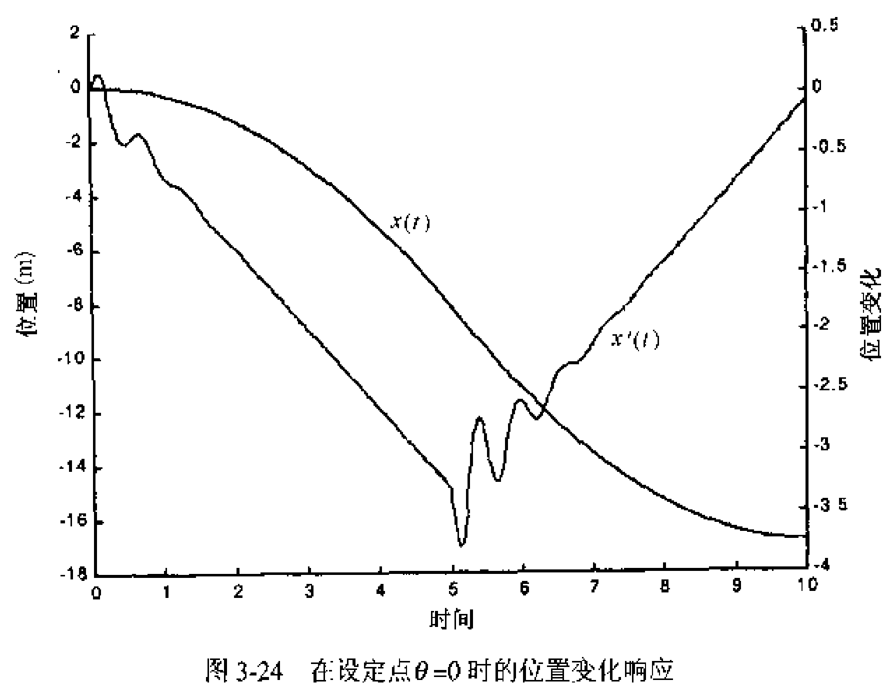


图 3-24　在设定点θ=0 时的位置变化响应

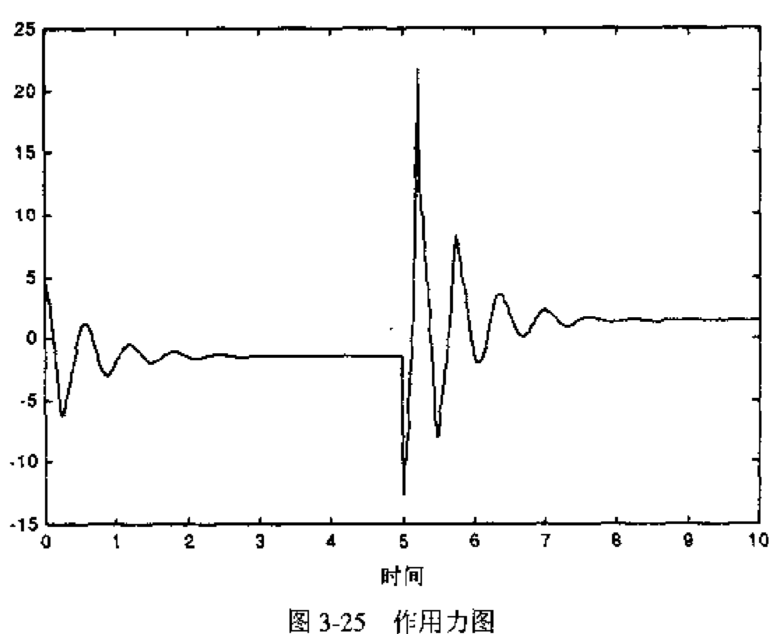


图 3-25　作用力图

3.5　位置 $x(t)$和角度$\theta(t)$的模糊控制

这是有关在小心保持悬挂的摆于垂直位置的同时调节 x。模糊控制需要 4 个输入：x、$\dot{x}$、θ、$\dot{\theta}$。相应的 SIMULINK 模型如图 3-26 所示。

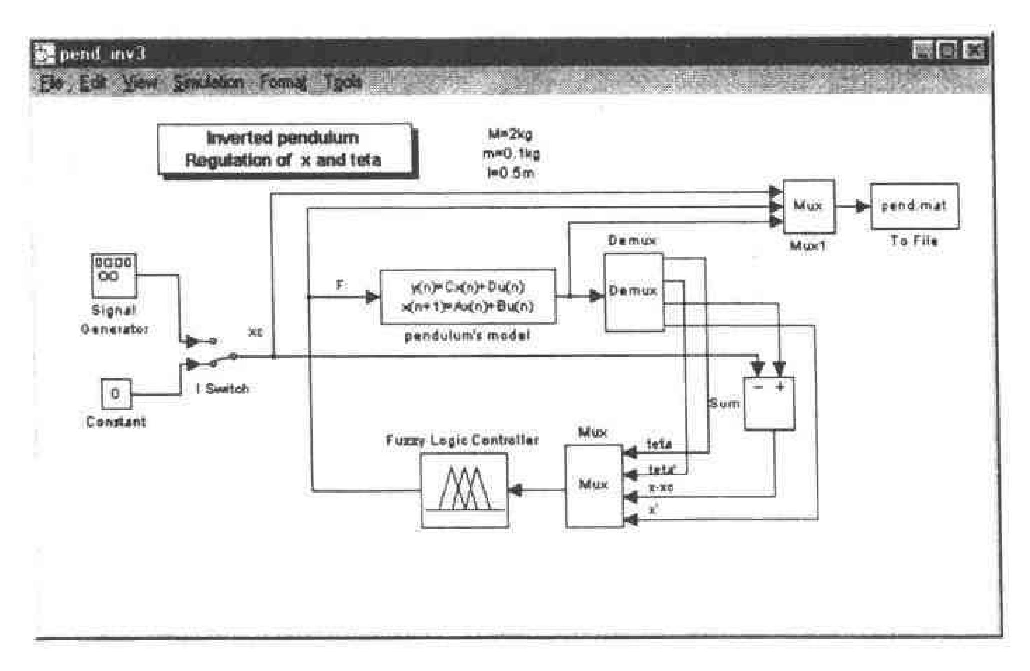


图 3-26 SIMULINK 模型

3.5.1 输入模糊化，隶属函数

现有 4 个输入，为每个输入固定 2 个规则，因此控制模糊规则为 16 个。用绘图编辑器产生的模糊控制保存在文件 fuz_x 中。为简化达到 DSP（数字信号处理），选择它的特性。变量 θ 和 $\dot{\theta}$ 的值域与前述一样，即

$$-0.3<\theta<0.3$$

$$-1<\dot{\theta}<1$$

位置 x 有 ±3 m 的范围变化，选择如下的位置 x 及其微分范围：

$$-3<x<3$$

$$-6<\dot{x}<6$$

对每个输入使用梯形模糊化 trapmf 函数。

如下命令行用绘图编辑器使隶属函数可视化。

fuzzy_teta_x.m

```
% loading the control fuzzy matrix
fismat1 = readfis('fuz_x');
% drawing inputs membership functions
figure(1)
plotmf(fismat1,'input',1), grid
title('\theta(t) Inputs membership functions')
xlabel('\theta(t) input')
ylabel('membership degree')
figure(2)
plotmf(fismat1,'input',2), grid
title('\theta''(t) Input membership functions ')
xlabel('\theta''(t) input')
ylabel('membership degree')
figure(3)
```

```
plotmf(fismat1,'input',3), grid
title('\theta''(t) Input membership functions')
title('x(t) membership functions')
xlabel('x(t) input')
ylabel('membership degree')
figure(4)
plotmf(fismat1,'input',4), grid
title(' x''(t) input membership functions')
xlabel('x''(t) input')
ylabel('membership degree')
```

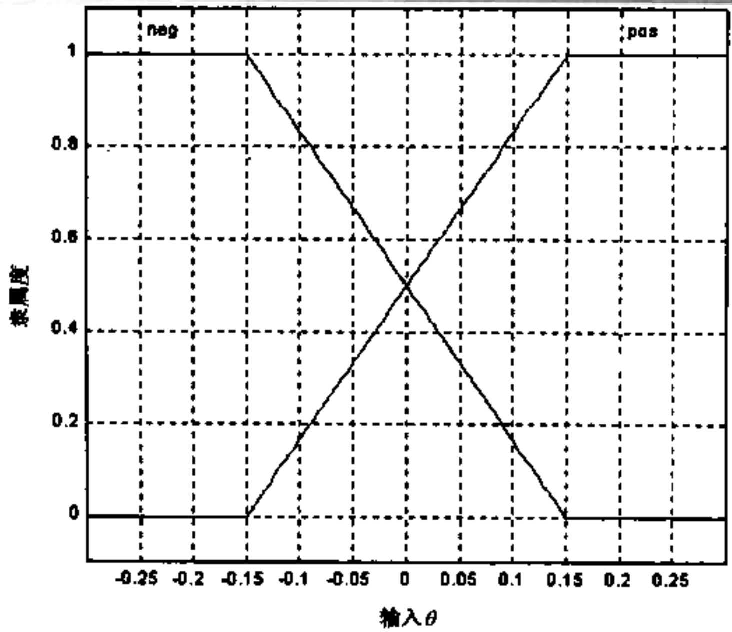


图 3-27 θ 输入隶属函数

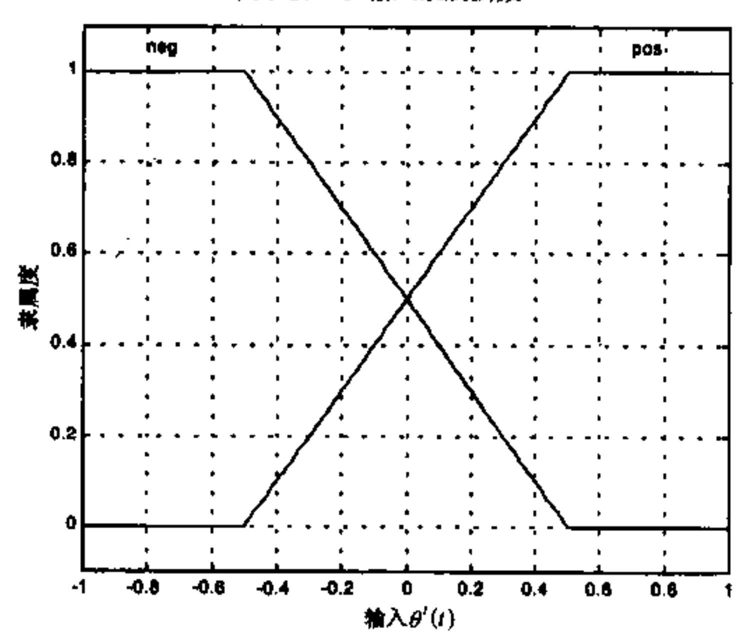


图 3-28 θ'(t)输入隶属函数

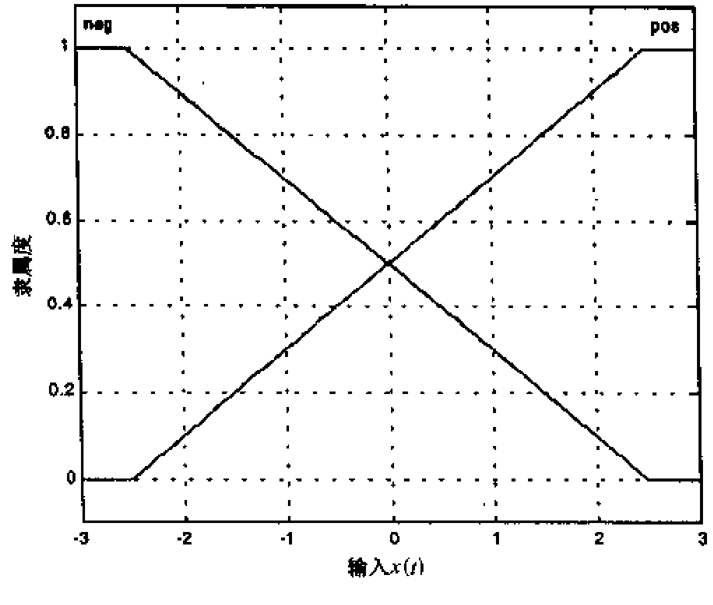


图 3-29 $x(t)$隶属函数

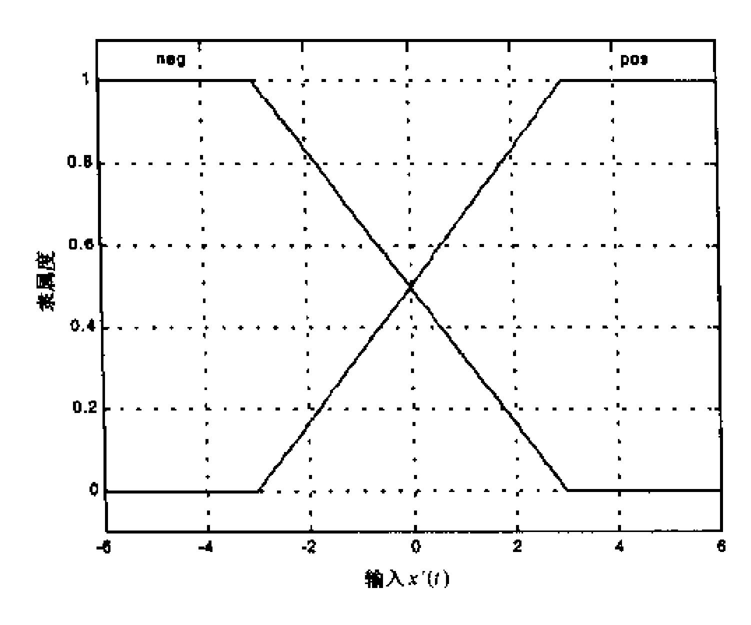


图 3-30 $x'(t)$输入隶属函数

使用 min-max 算符。因为模糊规则的数量较多，所以选用具有单独类型输出函数的 Sugeno 方法。通过使用重心方法非模糊来简化计算操作。使用这种方法，其含意是简单乘法和集合包括了所有的单点。16 个单点一般在输出变量 f（限制在 ±10N）定义范围内分配。

3.5.2 推理规则定义，非模糊化

与输入和有关单点输出相连的 16 个组合相对应的 16 个推理规则在下面推论表中表示。

力f（θ θ \ x x）	负 负	负 正	正 负	正 正
负 负	f1	f2	f3	f4
负 正	f5	f6	f7	f8
正 负	f9	f10	f11	f12
正 正	f13	f14	f15	f16

使用绘图编辑器的口头模式的规则编辑如下（见图 3-31）。

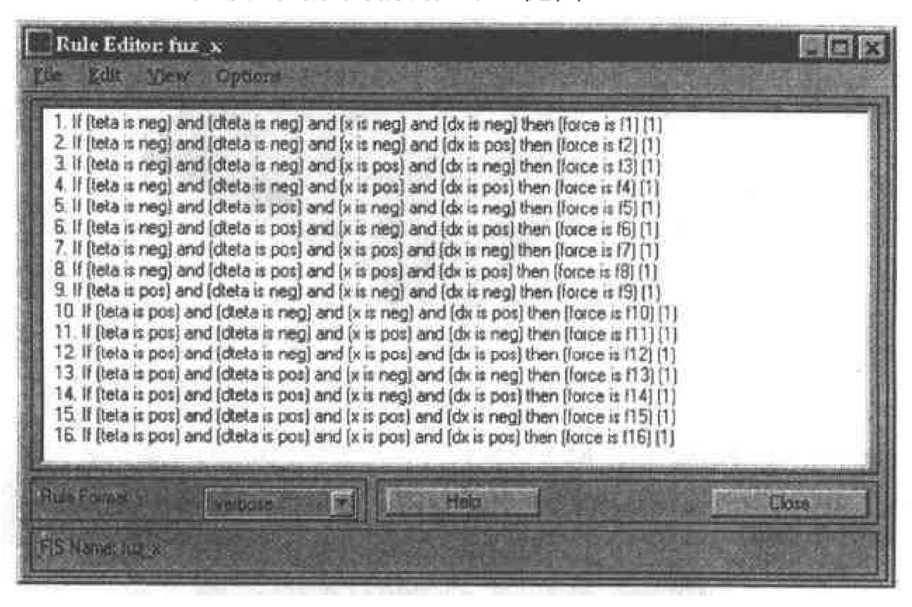


图 3-31 绘图编辑器

下面的命令行可显示表面图。

```
%Surface representation
figure (5) ,gensurf (fismat1, 1:2, 1)
title (`graph of angle surface `)
figure (6), gensurf(fismat1, 3:4,1)
title (`x position surface`)
```

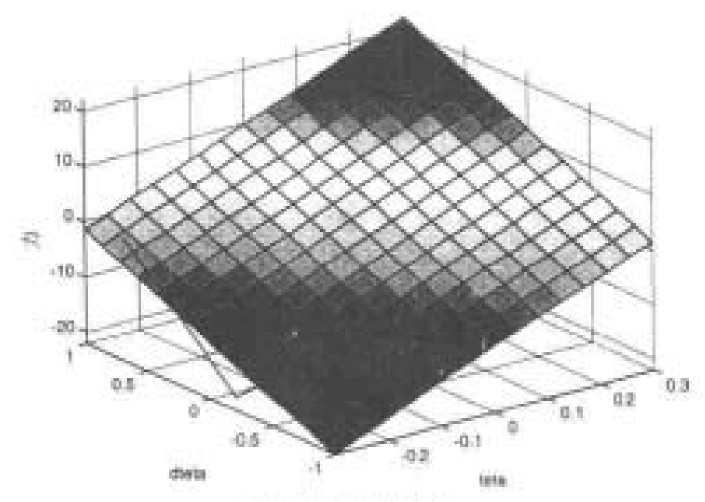


图 3-32 角表面图

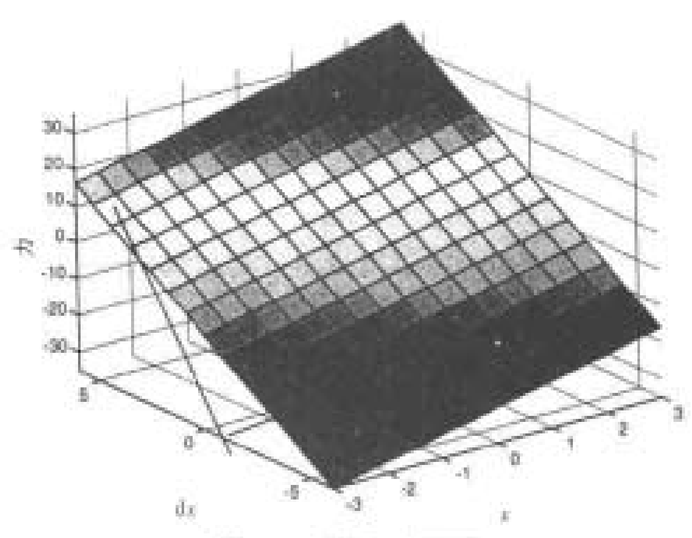


图 3-33 位置 x 表面图

在 ruleview 内的控制检测给出如下结果：

```
%Fuzzy controller test
ruleview (fismat1);
```

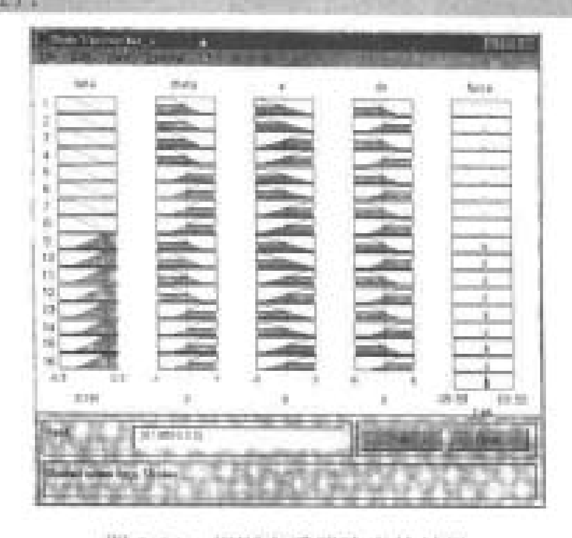


图 3-34 规则查看器给出的结果

此例中，控制与摆倾斜 0.191 rad 的作用产生 7.46N 的力。

3.5.3 获得模糊控制

含有 fuz_x 模糊矩阵的控制器矩阵称为 SIMULINK“模糊控制器”块的一个参数。

下面的文件 pendul_inv5.m 呈现的是根据位置设定点 $x_c(t)$ 的过程响应。起初，设定点为零，可以证明由如下矢量表示的初始状态的过程响应：

$$X_0 = \begin{bmatrix} 0.1 \\ 0 \\ -0.5 \\ 0 \end{bmatrix}$$

pendul_inv5.m

```
% Inverted pendulum
% x and teta regulation
% signals recuperation
load pend.mat
t = signals(1,:);
xc = signals(2,:);
f = signals(3,:);
teta = signals(4,:);
dteta = signals(5,:);
x = signals(6,:);
dx = signals(7,:);

% F(t) signal control drawing
figure(1)
plot(t,f), grid
title('f(t) signal control in N')
xlabel('Time')
gtext('\theta initial = 0.1 rd').
gtext('x initial = -0.5 m')

% angle drawing
figure(2)
hf = line(t,teta(:))
xlabel('Time')
ylabel('Angle in rd')
axet = axes('Position',get(gca,'Position'),...
          'XAxisLocation','bottom',...
          'YAxisLocation','right','Color','none',...
          'XColor','k','YColor','k');
ht = line(t,dteta,'color','r','parent',axet);
ylabel('angle variation in rd/s')
title('Response to the set-point x = 0')
gtext('\theta(t)'),gtext('\theta''(t)')
gtext('initial \theta = 0.1 rd')
gtext('x initial = -0.5 m')
```

```
% position drawing
figure(3)
hf = line(t,x(:));
xlabel('Time')
ylabel('Position in m')
axet = axes('Position',get(gca,'Position'),...
        'XAxisLocation','bottom',...
        'YAxisLocation','right','Color','none',...
        'XColor','k','YColor','k');
ht = line(t,dx,'color','r','parent',axet);
ylabel('position variation')
title('Response to the set-point x = 0')
gtext('x(t)'), gtext('x''(t)')
gtext('initial \theta = 0.1 rd')
gtext('initial x = -0.5 m')
```

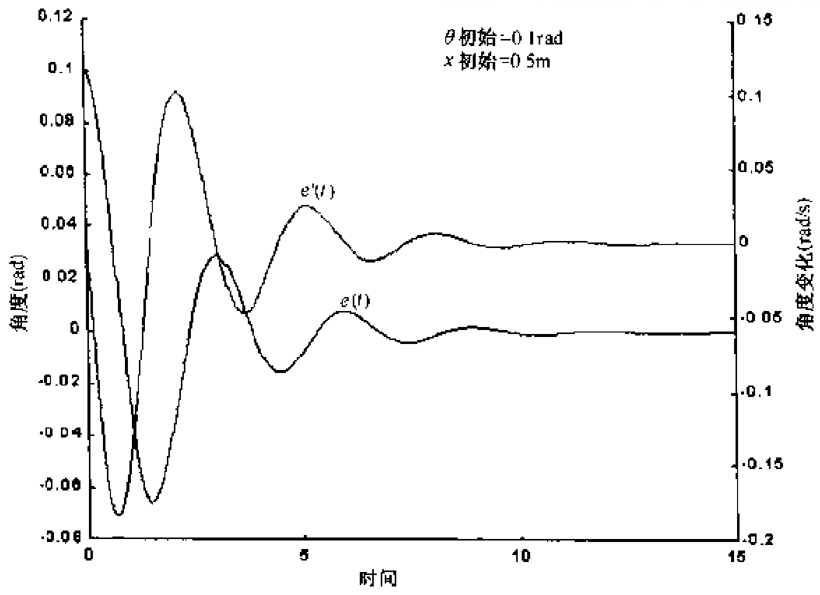


图 3-35　在设定点 x=0 的角度变化响应

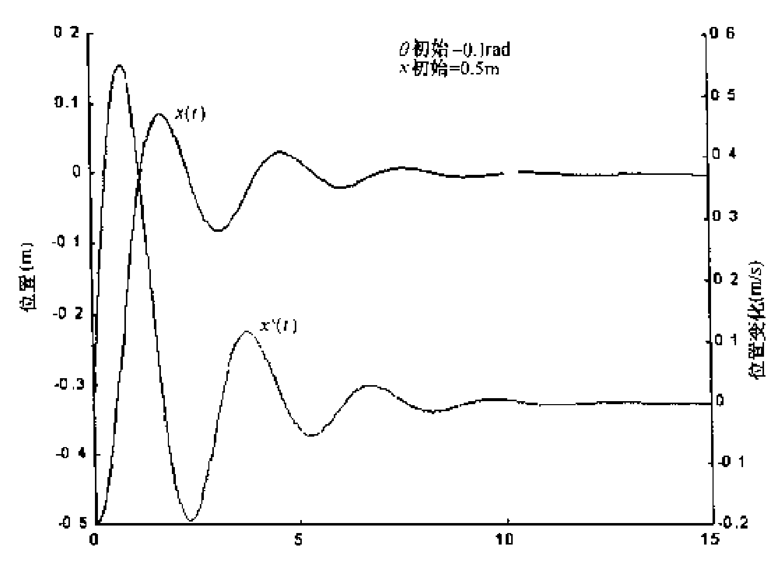


图 3-36　在设定点 x=0 的位置变化响应

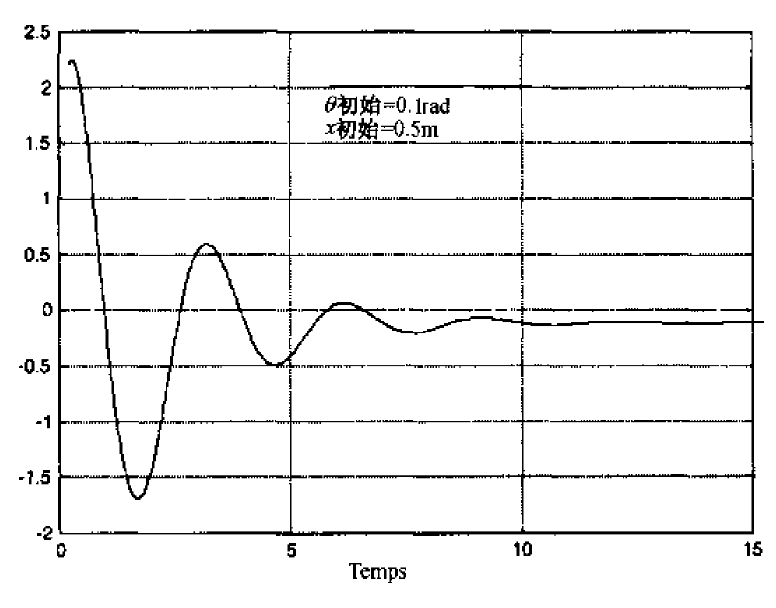


图 3-37　控制力 $f(t)$

若初始状态为θ=0.1rad、x=−0.5m，为将车返回 x=0、摆至θ=0 的位置，两者都需要一个正向力。事实上，摆必须转到左边，小车到右边。加一正力可满足这两个要求，可在前面的图中证明这一点。在如下初始状态下检查过程响应：

$$X_0 = [0.1 \quad 0 \quad 0.5 \quad 0]^T$$

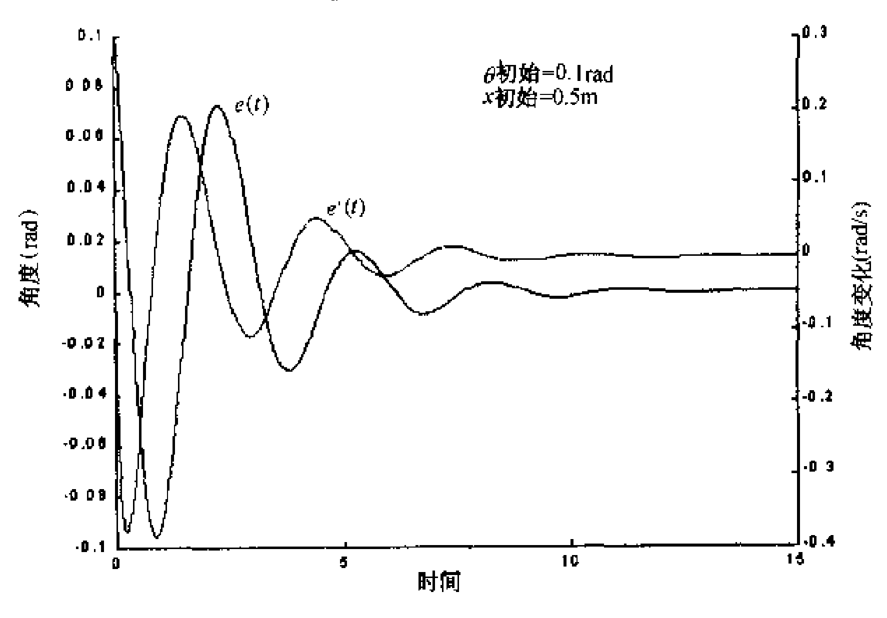


图 3-38　在设定点 x=0 的角度变化响应

根据所加的力，初始状态自我调节。为使摆为直，力必须是正的。而位置 x 要返回 0，则需要一个负力。

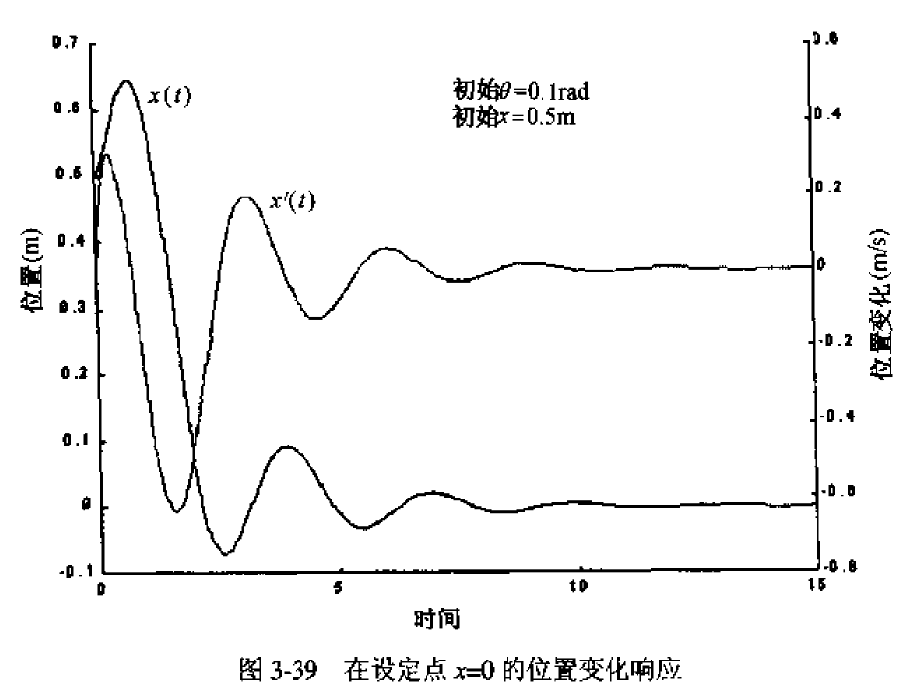


图 3-39 在设定点 x=0 的位置变化响应

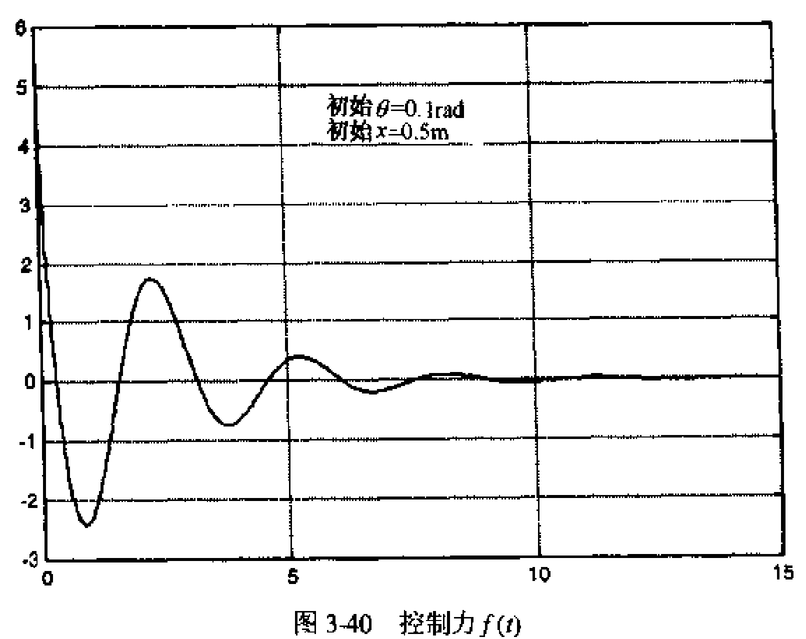


图 3-40 控制力 $f(t)$

可以看到，控制器将尽很大努力使角θ反向，并使小车朝理想位置移动一点。然后控制能将小车和摆同时到达理想位置。

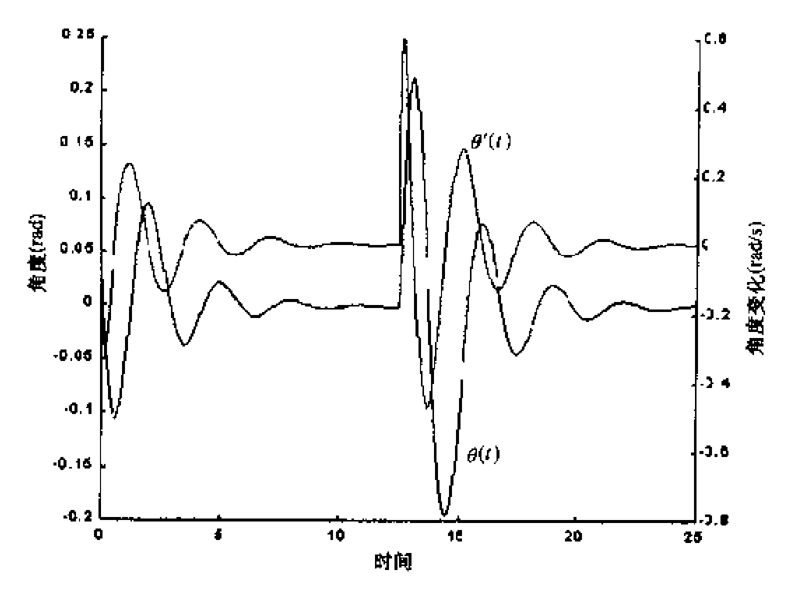


图 3-41 在设定点的角度变化响应

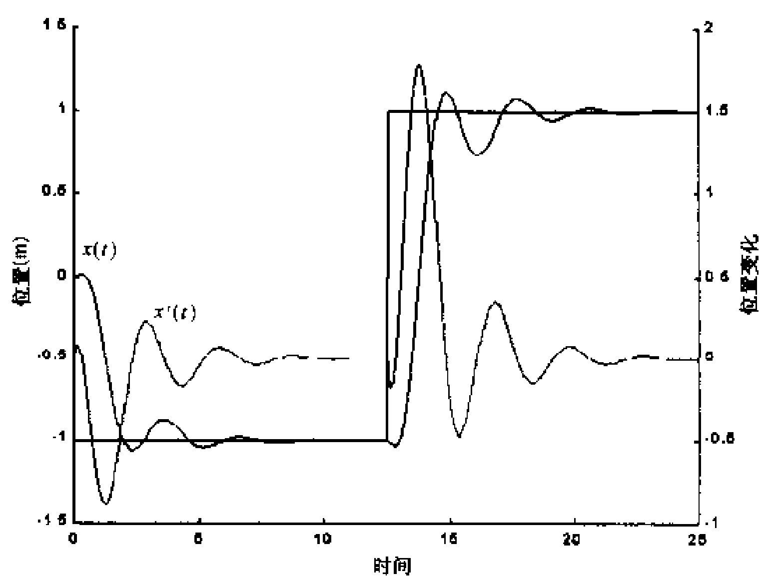


图 3-42 在设定点的位置变化响应

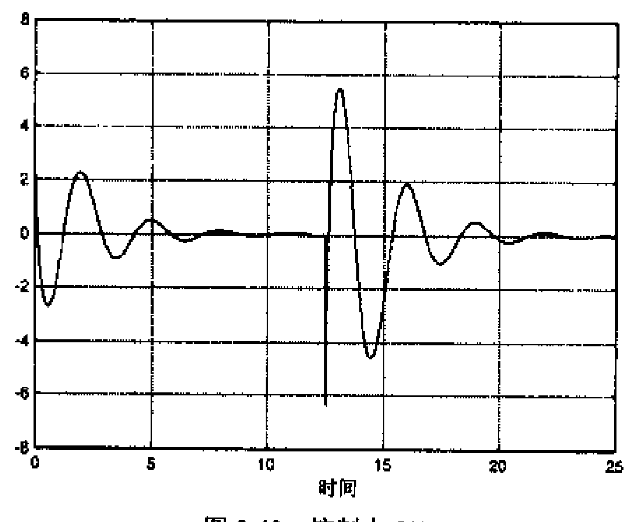


图 3-43 控制力 $f(t)$

我们看到，在每个新设定点值，为使摆在小车位移时倾斜，应加上脉冲。角度偏移至少 10° 左右，这对应于小角度的线性近似。

3.6 系统的图解显示

要实现车和反转摆的图解显示，需使用 S 函数（详见附录），这个函数称为 animat.m。SIMULINK 模型修改如下：

- 加 S 函数块，让输入 x 和θ传给它作为参数，于是

$$u=[\theta(k)\ \ x(k)]$$

- 为在过程控制信号上添加随机干扰，加上 uniform random number 块。

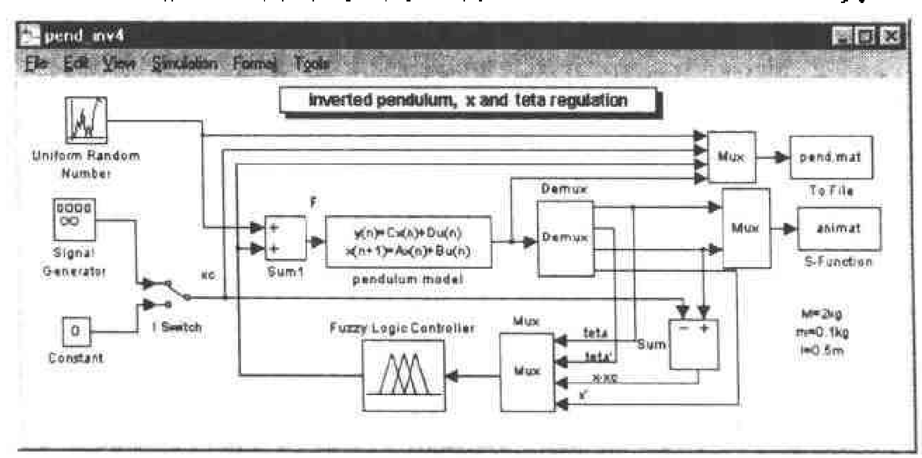


图 3-44 SIMULINK 模型

车和摆的移动轨迹如图 3-45 所示。这里固定直角参考坐标，X 轴范围为±50，Y 轴范围为-10~+90。

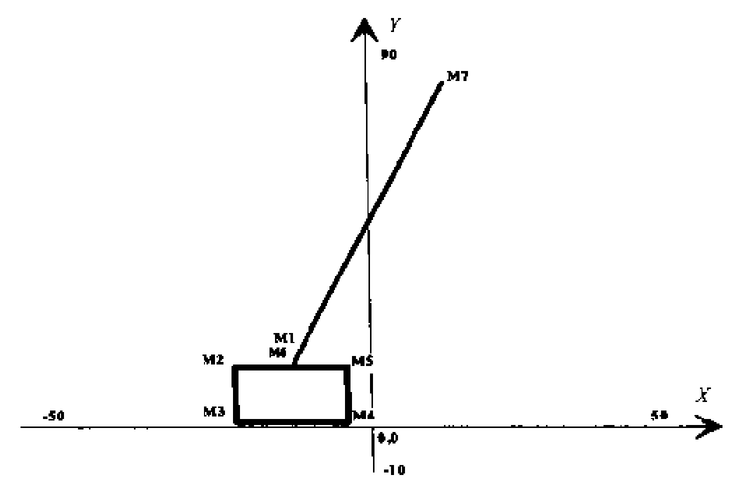


图 3-45 车和摆的移动轨迹图

移动轨迹图由 2 个矢量表示：X 轴的 Mx 坐标、Y 轴的 My 坐标。

animat.m file

```
% Graphical simulation of the inverted pendulum
function [sys,x0,str,ts] = animat(t, x, u, flag,Te)
```

```
global pendulum
switch flag

% Initialisation stage
case 0

     animinit('Pendulum animation');
     pendulum = findobj('Type','figure','Name','Pendulum animation');
     axis([-50 50 -10 90])
     hold on

     % drawing the fixed image (ground)
     plot([-50 50],[0 0],'blue',[-50:50;-50:50],[-5 0],'blue','LineWidth',1);

     % mobile vectors initialisation of mobile images
     % ( cart + inverted pendulum )
     Mx = [0 -5 -5 5 5 0 0];
     My = [3 3 0 0 3 3 53];

     % drawing the mobile image
     hndl = plot(Mx,My,'blue','erasemode','xor','linewidth',2);
     set(gca,'UserData',hndl);
     [sys,x0,str,ts] = initialisation(Te);
     % Input vectors initialisation stage for up-dating
     % drawing
case 2
     % If the figure exists, we recover its pointer
    if any(get(0,'Children') == pendulum),
     if strcmp(get(pendulum,'Name'),'Pendulum animation'),
        set(0,'currentfigure',pendulum);
        hndl = get(gca,'UserData');

        % Reading x(k) input
        Cx = 40*u(2);
        % reading input teta(k) and polar co-ordinates
        %transformation
        [tetay,tetax] = pol2cart(u(1),50);
        Px = Cx+tetax;
% vectors Mx and My calculation
        Mx = [Cx Cx-5 Cx-5 Cx+5 Cx+5 Cx Px];
        My = [3 3 0 0 3 3 3+tetay];

% drawing up-dating
        set(hndl,'Xdata',Mx,'Ydata',My);
        drawnow
      end
     end
     sys = [];
% non-used stages
case {1,3,4,9}
   sys = [];
```

```
otherwise
   error(['Unhandled flag = ',num2str(flag)])
end

% Initialisation function
  function [sys,x0,str,ts] = Initialisation(Te)
  sizes = simsizes;          % Structure of type simsizes
  sizes.NumContStates = 0; % Number of continuous states
  sizes.NumDiscStates = 0; % Number of discrete states
  sizes.NumOutputs = 0;     % Number of outputs
  sizes.NumInputs = 2;      % Number of inputs
  sizes.DirFeedthrough = 0;% Non using of D
  sizes.NumSampleTimes = 1;% Number of ts lines
  sys = simsizes(sizes);    % Up-dating of the system vector
  x0 = [];                  % non specified initial state
  str = [];                 % empty chain  (M function)
  ts = [Te 0];              % sampling period = Te
```

◆ 得到的图解显示（见图 3-46）

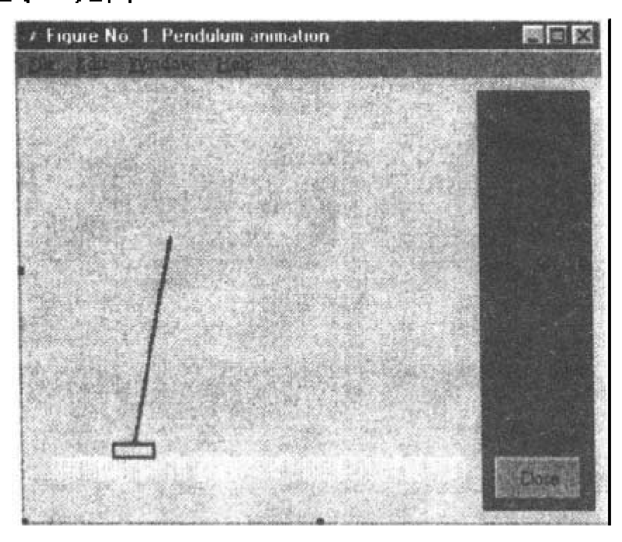


图 3-46　得到的图解显示

下面的图呈现的是根据持续 10 s 的随机干扰 $b(t)$和位置设定点 X_c 为 0 的调节模式下的过程输出。

pendul_inv6.m file

```
% Inverted pendulum
% x and teta regulation

% signals recuperation
load pend.mat
t = signals(1,:);
b = signals(2,:);
f = signals(4,:);
teta = signals(5,:);
dteta = signals(6,:);
x = signals(7,:);
```

```
dx = signals(8,:);

% disturbance and F(t) control signals drawing
figure(1)
plot(t,b,'r:')
hold on
plot(t,f)
hold off
title('disturbance and F(t) control in N')
xlabel('Time')
gtext('fp(t)')
gtext('f(t)')

% x position drawing
figure(2)
plot(t,teta)
title('Angle \theta(t) in rd'), xlabel('Time')

% theta angle drawing
figure(3), plot(t,x)
title(' x(t) position in m'), xlabel('Time')

% outputs dx and dteta drawing
figure(4);
hf=line(t,dteta(:));
xlabel('Time')
ylabel('angle derivative in rd/s')
axet=axes('Position',get(gca,'Position'),...
        'XAxisLocation','bottom',...
        'YAxisLocation','right','Color','none',...
        'XColor','k','YColor','k');
ht=line(t,dx,'color','r','parent',axet)
ylabel('position derivative in m/s')
title('x(t) and \theta(t) regulation')
gtext('\theta''(t)\rightarrow'), gtext('\leftarrowx''(t)')
```

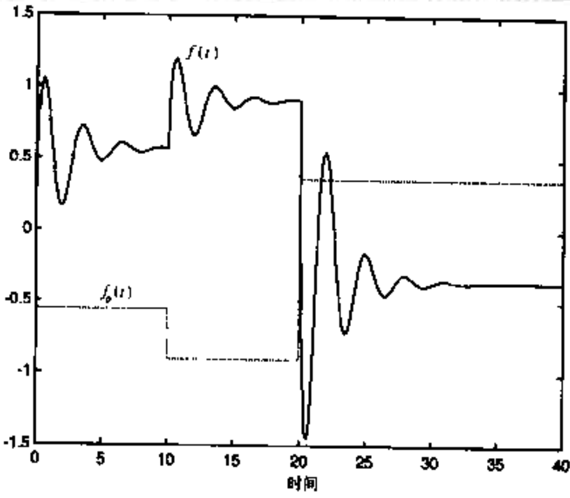


图 3-47　干扰$f_p(t)$和控制 $F(t)$

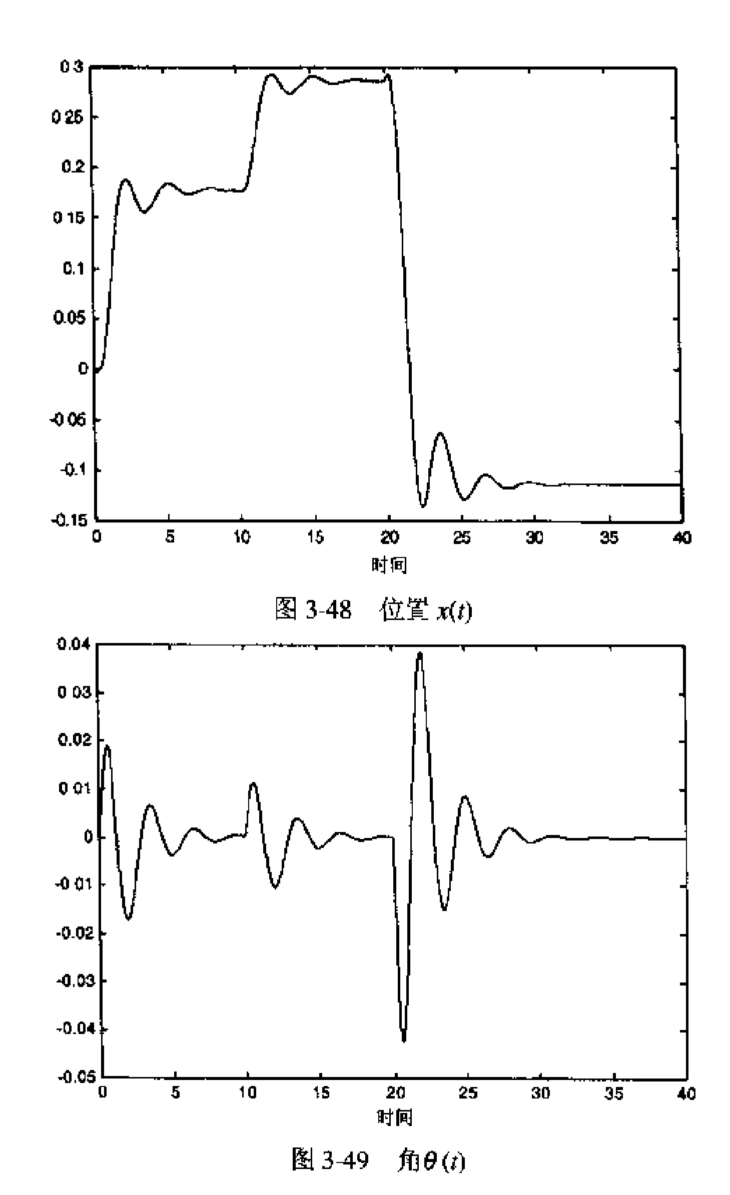


图 3-48　位置 $x(t)$

图 3-49　角 $\theta(t)$

可以看到，对悬挂摆的干扰效果很快就被补偿，这与它对小车干扰形成很明显的对比。

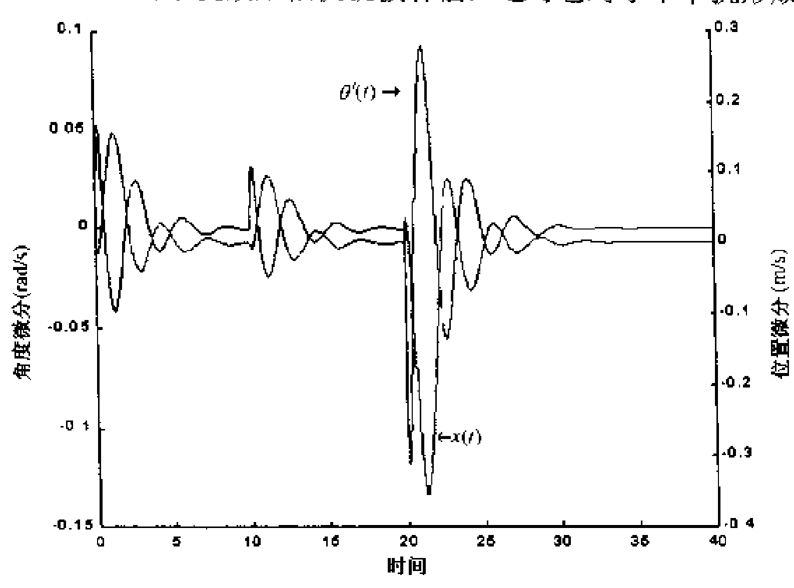


图 3-50　位置 x 和角度调节图

应用4 烤 箱 控 制

本应用是控制一个通风烤箱的温度。热量由热电阻产生，由功率放大器产生的电压 V_c 控制。温度由放在测量孔中的热电偶测量，仪表放大器产生电压 V_m，显示温度 θ_m。在烤箱温度范围内，假定传感器和仪表放大器装置是线性的（见图 4-1）。

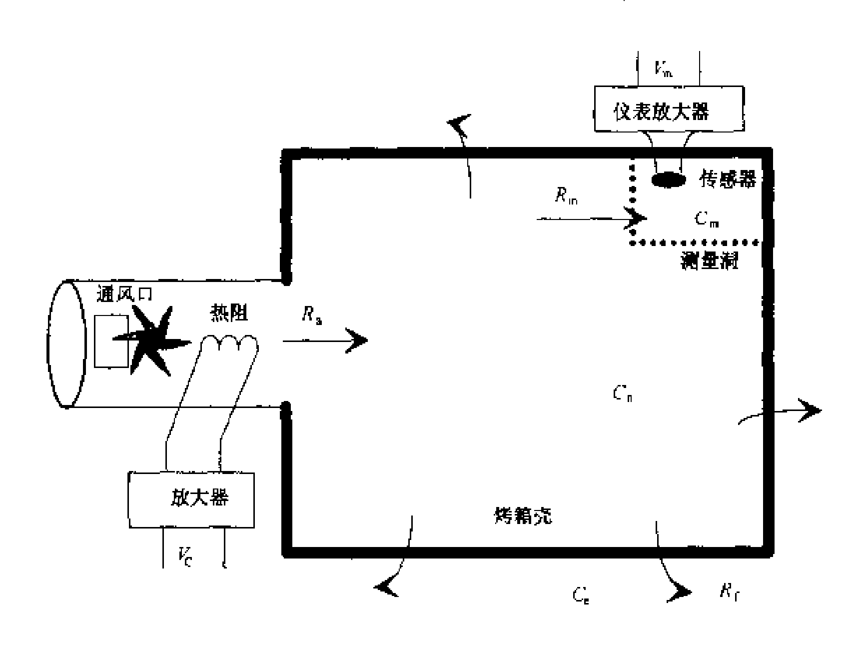


图 4-1 烤箱结构图

此过程包括的参数如下：

- $Q = k_t V_c$ 产生的热量；
- R_a 减少导管热向机壳传播的热电阻；
- C_a 机壳的热容；
- R_m 降低测量洞中的烤箱热循环热电阻；
- C_m 测量洞的热容；
- R_f 降低朝烤箱外的热循环的漏泄电阻；
- C_e 烤箱外的热容，认为很大；
- θ_a、θ_m、θ_e 分别表示烤箱机壳、测量洞和烤箱外的温度。

4.1 烤箱模型

等价的电路图 4-2 前面的热力学系统可用如下方程描述：

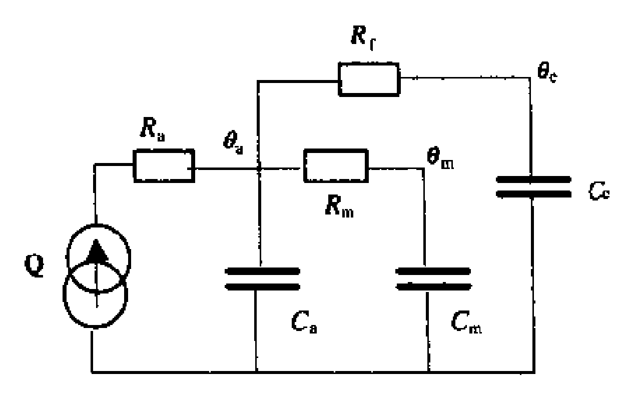


图 4-2　等价电路图

$$Q = C_a \frac{d\theta_a}{dt} + C_m \frac{d\theta_m}{dt} + \frac{\theta_a - \theta_e}{R_f}$$

其中

$$\theta_m = \theta_a - R_m C_m \frac{d\theta_m}{dt}$$

对上式使用拉普拉斯变换，得到根据 Q、θ_e 和系统参数的关于 θ_a 的表达式。经过一些计算后，也就是

$$\theta_a = \left(Q + \frac{\theta_e}{R_f}\right)\left(\frac{R_f(1+R_m C_m p)}{1+(R_m C_m + R_f C_m + R_f C_a)p + R_f R_m C_m C_a p^2}\right)$$

测量温度和烤箱机壳温度的关系如下：

$$\frac{\theta_m}{\theta_a} = \frac{1}{1+R_m C_m p}$$

考虑这些方程，完整的过程用如下函数方框图 4-3 描述。

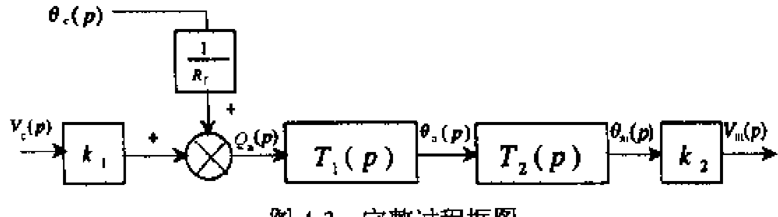


图 4-3　完整过程框图

其中：

$$T_1(p) = \frac{R_f(1+R_m C_m p)}{1+(R_m C_m + R_f C_m + R_f C_a)p + R_f R_m C_m C_a p}$$

$$T_2(p) = \frac{1}{1+R_m C_m p}$$

数字应用

烤箱参数取如下的值：

- R_a=0.01℃/W
- R_m=3℃/W
- R_f=0.1℃/W
- Q_{max}=5 000 W

- C_a=5 000 J/℃
- C_m=10 J/℃
- $C_e=\infty$
- θ_e =20℃
- k_1=100 W/V
- k_2=0.1 V/℃

经过计算后，也就是

$$T_1(p)=\frac{0.1+3p}{1+531p+1500p^2}$$

$$T_2(p)=\frac{1}{1+30p}$$

由文件 oven_mod.m 得到模拟模型。

oven_mod.m file

```
% Process parameters
% gains
k1 = 100; k2 = 0.1;

% thermal resistances
Ra = 0.01; Rm = 3; Rf = 0.1;
% thermal capacities
Ca = 5000; Cm = 10;

% time constants
Tau_m = Rm*Cm;
Tau_fm = Rf*Cm;
Tau_fa = Rf*Ca;

% analogical model
sys = tf(k1*k2*Rf, [Tau_m*Tau_fa (Tau_m+Tau_fm+Tau_fa) 1]);

% numerator and denominator of the transfer functions
[num, den] = tfdata(sys,'v');
disp(' transfer function ')
printsys(num, den)
transfer function
num/den=
     1
-------------
15000s^2+531s+1
```

这个模拟模型可在如下 SIMULINK 文件 oven_1.mdl 的帮助下进行检测（见图 4-4）。

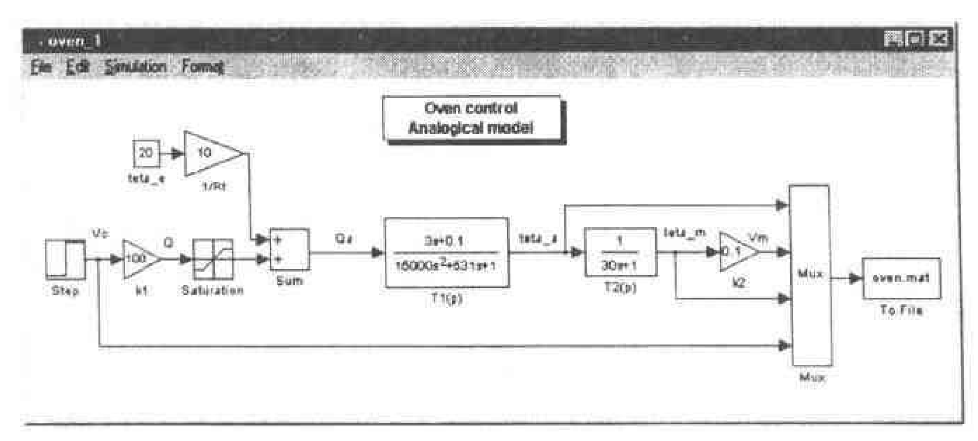


图 4-4　烤箱控制模拟模型

饱和限制产生的热量最大为 5 000 W。文件 signal_ovenl.m 呈现的是在 1 000 W 功率下的烤箱反应。

signal_ovenl.m file

```
load oven.mat
t = signal(1, :);
Qa = signal(2, :);
Vm = signal(3, :);
dVm = signal(4, :);
Vc = signal(5, :);
figure(1)
plot(t, Vc)
hold on
plot(t, Vm, 'r'), hold off
title(' Vc order and Vm measure ')
xlabel('Time')
ylabel('Voltage in V')
figure(2)
plot(t, teta_a),hold on
plot(t,teta_m,'r'),hold off
xlabel('Time'),ylabel('Temperature in •C')
gtext('\thetaa\rightarrow')
gtext('\leftarrow\thetam')
```

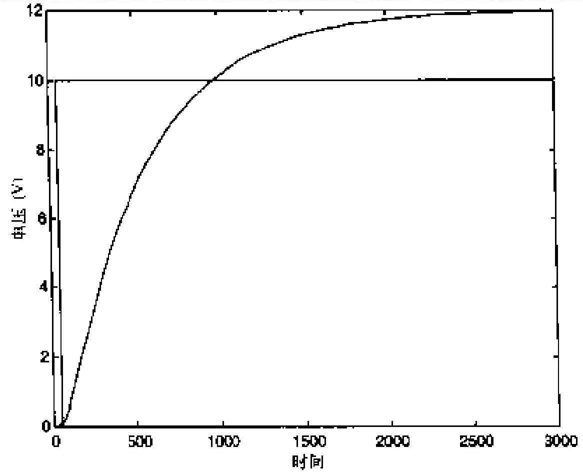


图 4-5　控制电压 V_c 和测量电压 V_m

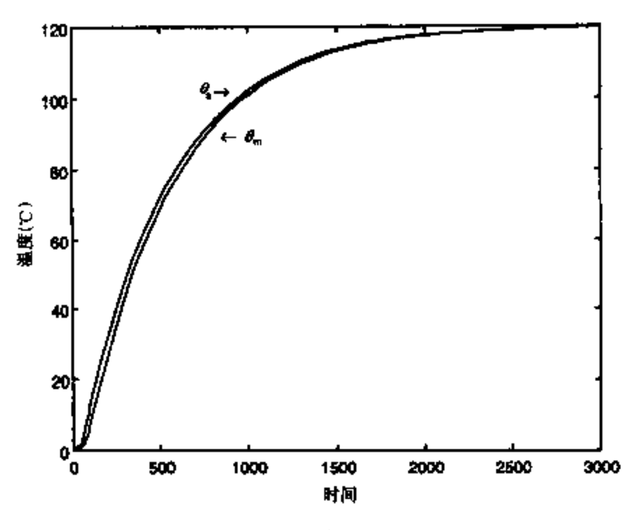


图 4-6　烤箱温度图

烤箱相当于一个时间常数接近 500 s 的一阶过程。在瞬间，测量温度 θ_m 很接近烤箱温度 θ_s。在稳定状态下，它们当然是相等的。

4.2　具有零极点补偿的积分控制

在 SIMULINK 文件 oven_controll.mdl 中实现了具有零极点补偿的积分控制模型。采样周期为 T_s=1 s（见图 4-7）。

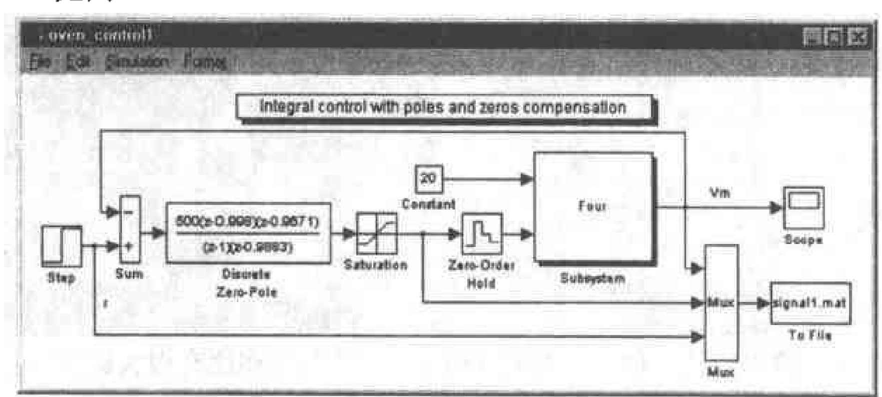


图 4-7　有零极点补偿的积分控制模型

模型由图 4-8 所示的子模型定义，这里描述它的 $T(p)$模拟传递函数。系统不同的信号保存在文件 signal1.mat 中。

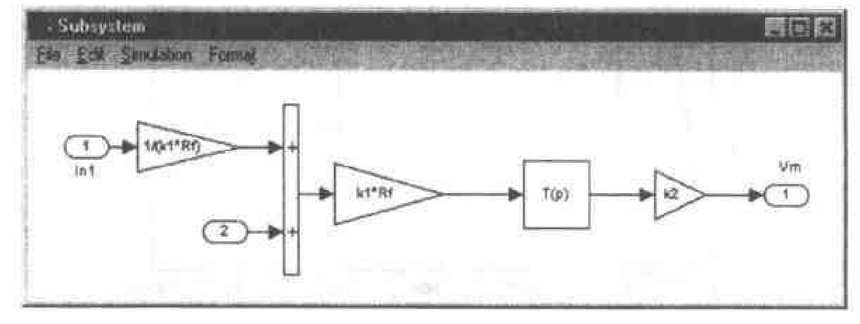


图 4-8　子系统图

测量电压 V_m 与控制电压 V_c 间的传递函数 $T(p)$如图 4-9 所示。

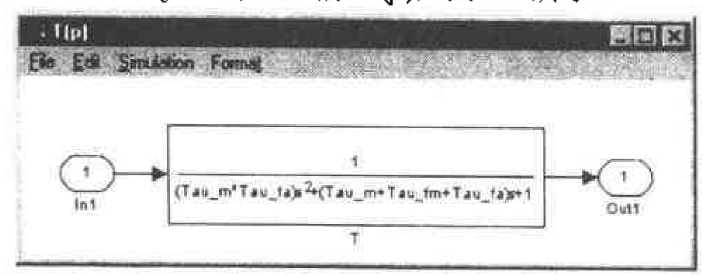


图 4-9 $T(p)$图

文件 signal_oven2.m 显示修正系统的信号。

signal_oven2.m file

```
% Integral control and compensation of poles and zeros
% different signals graph

close all
load signal1.mat
t = signal(1,:);
Vm = signal(2,:);    % sensor voltage
Vc = signal(3,:);   % control signal
r = signal(4,:);     % order signal
% display of the measure and of the order
figure(1)
plot(t,Vm), hold on
plot(t,r), grid
title(' Vm measuring voltage and r order')
xlabel('samples')

% display of the control signal
figure(2)
stairs(t,Vc)
title(' Vc control signal ')
xlabel('samples')
```

我们观察到有输出超调，但响应时间 150 s 与开环系统相比有很大提高。

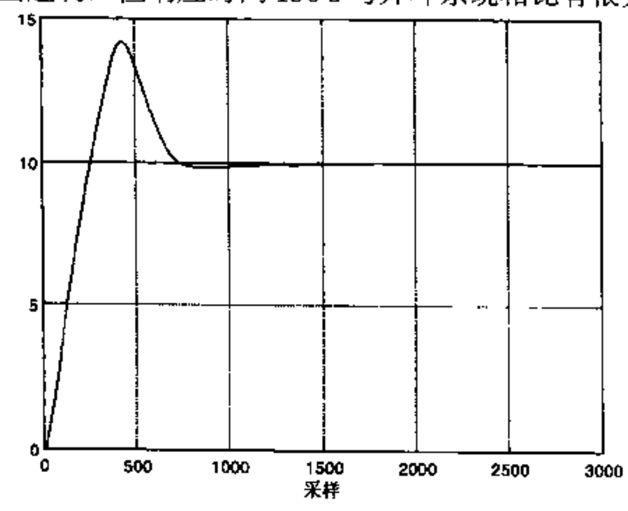


图 4-10 测量电压 V_m 和阶数 r

为得到这个很小的响应时间，控制在 0 V~50 V 间的暂态振荡，然后在稳态稳定。如果降低积分增益，就可减少导致增加响应时间的超调（见图 4-11）。

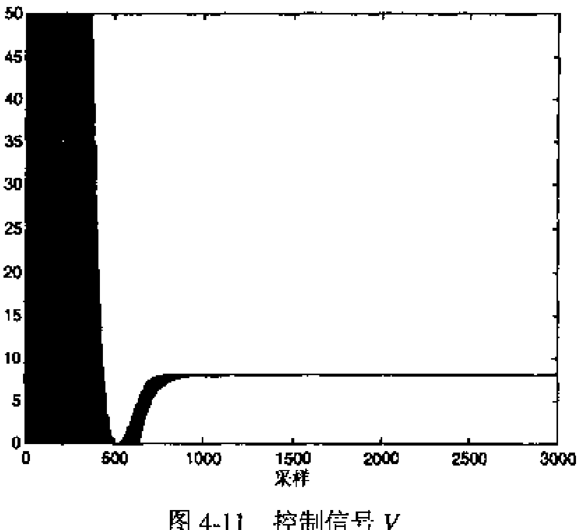


图 4-11　控制信号 V_c

4.3　烤箱的离散状态表示

为得到传递函数 $T(p)=\dfrac{V_m(p)}{V_a(p)}$ 的状态表达式，修改烤箱函数框图如下，这里 $V_a(p)$反映烤箱机壳的温度。

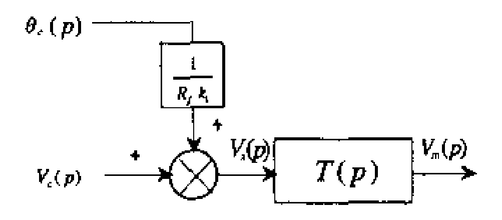


图 4-12　修改后的烤箱函数框图

其中

$$T(p)=\frac{k_1k_2R_f}{1+(R_mC_m+R_fC_m+R_fC_a)p+R_fR_mC_mC_ap^2}$$

$T(p)$是二阶函数，因此状态表达式需要 2 个状态变量：

$$\begin{cases}x_1(t)=V_m(t)\\ x_2(t)=\dot{x}_1(t)\end{cases}$$

如果如下修改 $T(p)$表达式：

$$T(p)=\frac{b_0}{p^2+a_1p+a_0}$$

得到对应于 $T(p)$传递函数的状态表达式：

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -a_0 & -a_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ b_0 \end{bmatrix} V_c$$

$$y = V_m = [1 \quad 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

考虑过程的时间常数，其被采样周期 1 s 所离散化。通过测量 V_m 来控制烤箱温度。为此，使用极点配置二阶系统：

- 固有频率 ω_n =0.005 rad/s；
- 阻尼系数 ξ =0.707。

文件 ovenl.m 决定了状态反馈矢量 K。

oven1.m file

```
% Oven control, state return
Rm = 3;
Rf = 0.1;
Ca = 5000;
Cm = 10;
k2 = 0.1;
k1 = 100;
Te = 1;

b0 = k1*k2*Rf/(Rf*Rm*Cm*Ca);
a0 = 1/(Rf*Rm*Cm*Ca);
a1 = 1/(Rm*Cm);

% Representation in the state space

A = [0  1 ; -a0 -a1];
B = [0 ;b0 ];
C = [1 0];
D = 0;

% discretisation of the state representation
sys = ss(A, B, C, D);
sysd = c2d(sys, Te, 'zoh');
[Ad, Bd, Cd, Dd, Ts, Td] = ssdata(sysd)

Ad =
    1.0000      0.9835
   -0.0001      0.9672
Bd =
   1.0e-004 *
          0.3297
          0.6557
cd = 1     0
Dd = 0
Ts = 1
Td = [ ]
```

状态反馈矢量的计算由 Ackerman 方法获得，这里 Com 代表过程命令矩阵，$\Phi(A)$ 为寻求的特征方程：

$$K = [0 \quad 1]\mathrm{Com}^{-1}\Phi(A)$$

在计算前有必要核对过程命令。

在修正过程中寻找的由多项式 $P(z)$所描述的二阶动力系统由下列方程得到：

$$P(z) = z^2 + p_1 z + p_2$$

$$p_1 = -2\mathrm{e}^{-\xi\omega_n T_e}\cos\left(\omega_n T_e\sqrt{1-\xi^2}\right)$$

$$p_2 = \mathrm{e}^{-2\xi\omega_n T_e}$$

建议通过 MATLAB 提供的多个函数来计算状态反馈系数。

◆ 使用 polyvalm 和 ctrb 控制

```
% process dynamics
m = sqrt(2)/2;
wn = 1/200;
p1 = -2*exp(-m*wn*Te)*cos(wn*Te*sqrt(1-m^2));
p2 = exp(-2*m*wn*Te);
p = [1 p1 p2];
% Ackermann method
phi = polyvalm(p, Ad);
comb = ctrb(Ad, Bd);
disp(['rang = 'num2str(rank(comb))]);
K = [ 0 1]*(inv(comb))*phi

K = -0.6201  -392.8495
```

使用“控制系统工具箱”的 acker 函数可得到相同的结果。

◆ 使用 acker 函数

```
pr = roots(p);
K1 = acker(Ad, Bd, pr)

K1= -0.6201  -392.8495
```

在单输入系统中，控制 place 给出同样的结果，暗示获得的极点位置的准确性。

◆ 使用 place 函数

```
K2 = place(Ad, Bd, pr)
Place: ndigita = 15

K2 = -0.6201  -392.8495
```

◆ 具有状态反馈的 SIMULINK 离散过程模型（见图 4-13）

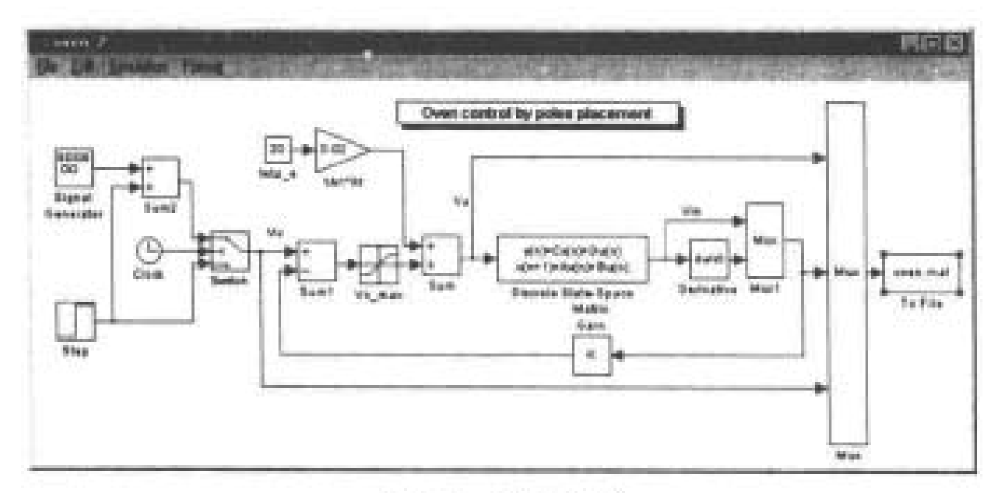


图 4-13　烤箱控制图

文件 signal_oven3.m 使我们能够表示在 100 ℃阶跃指令持续 2000s 下得到的信号。

signal_oven3.m file

```
% Oven control
% Poles placement by state return
load oven.mat
t = signal(1, :);
Va = signal(2, :);
Vm = signal(3, :);
dVm = signal(4, :);
Vc = signal(5, :);

% Tracing of the Vc order and the Vm measure
figure(1)
plot(t, Vc)
hold on
plot(t, Vm, 'r')
hold off
title('Vc order and Vm measure')
xlabel('Time'), ylabel('Voltage in V')
gtext('\leftarrowVc')
gtext('\leftarrowVm')

% plotting of the heat quantity
figure(2)
plot(t, 100*Va)
title('Heat quantity')
xlabel('Time')
ylabel('Power in W')
```

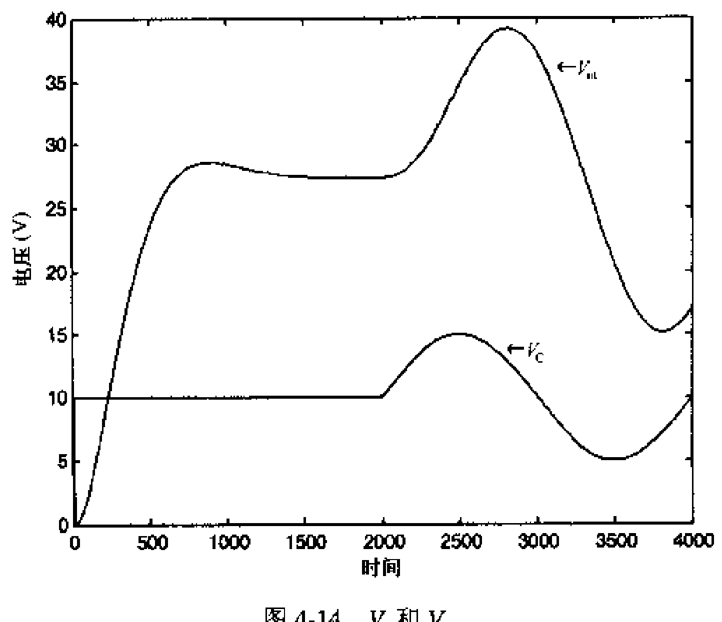


图 4-14 V_c和 V_m

可得到寻找的动态特性，但是只要过程不包括积分，位置误差就存在于全过程。

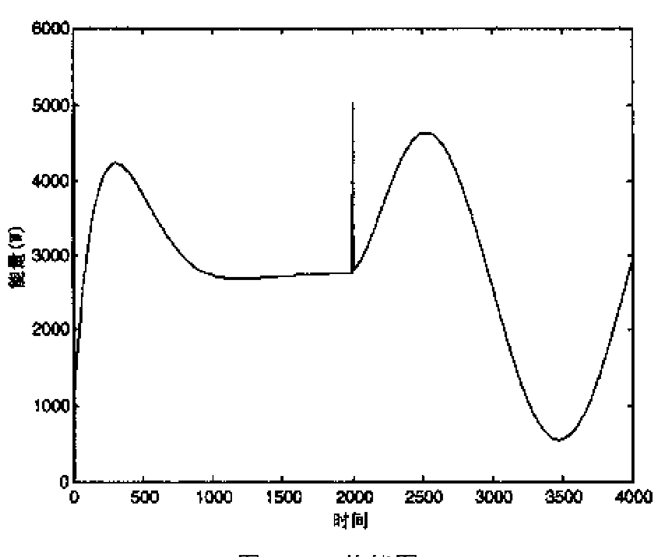


图 4-15 热能图

当指令突然变化时，控制在 5 000 W 时饱和。

4.4 具有积分的状态反馈控制

为消除位置误差，可修改控制以在前向中加入积分。修正过程的函数框图如图 4-16 所示。

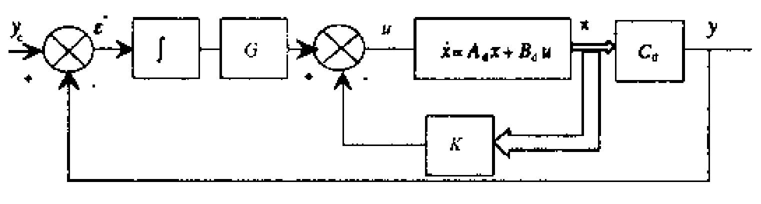


图 4-16 加入积分后的过程框图

这里：

$$u(k) = -Kx(k) + g\varepsilon(k)$$

$$\varepsilon(k) = \frac{1}{1-z^{-1}}\left[y_c(k) - y(k)\right]$$

也就是

$$\varepsilon(k+1) = \varepsilon(k) + y_c(k+1) - y(k+1)$$

计算后，得到

$$\begin{cases} x(k+1) = (A_d - B_d K)x(k) + B_d g\varepsilon(k) \\ \varepsilon(k+1) = (C_d B_d K - C_d A_d)x(k) + (1 - C_d B_d g)\varepsilon(k) + y_c(k+1) \end{cases}$$

系统可用矩阵形式表示为

$$\begin{bmatrix} x(k+1) \\ \varepsilon(k+1) \end{bmatrix} = \begin{bmatrix} A_d - B_d K & B_d g \\ C_d B_d K - C_d A_d & 1 - C_d B_d g \end{bmatrix} \begin{bmatrix} x(k) \\ \varepsilon(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} y_c(k+1)$$

$$y(k) = \begin{bmatrix} C_d & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ \varepsilon(k) \end{bmatrix}$$

如果考虑在单位阶跃下的系统响应，那么

$$\begin{cases} y_c(t) = 1 \\ y(\infty) = y_c(\infty) = 1 \end{cases}$$

如果表示如下误差矢量

$$\begin{bmatrix} x_{err}(k) \\ \varepsilon_{err}(k) \end{bmatrix} = \begin{bmatrix} x(k) - x(\infty) \\ \varepsilon(k) - \varepsilon(\infty) \end{bmatrix}$$

那么

$$\begin{bmatrix} x_{err}(k+1) \\ \varepsilon_{err}(k+1) \end{bmatrix} = \begin{bmatrix} A_d - B_d K & B_d g \\ C_d B_d K - C_d A_d & 1 - C_d B_d g \end{bmatrix} \begin{bmatrix} x_{err}(k) \\ \varepsilon_{err}(k) \end{bmatrix}$$

分解上式以使使具有积分的状态反馈系数矢量出现：

$$\begin{bmatrix} x_{err}(k+1) \\ \varepsilon_{err}(k+1) \end{bmatrix} = \begin{bmatrix} A_d & 0 \\ -C_d A_d & 1 \end{bmatrix} \begin{bmatrix} x_{err}(k) \\ \varepsilon_{err}(k) \end{bmatrix} + \begin{bmatrix} -B_d K & B_d g \\ C_d B_d K & -C_d B_d g \end{bmatrix} \begin{bmatrix} x_{err}(k) \\ \varepsilon_{err}(k) \end{bmatrix}$$

即

$$\begin{bmatrix} x_{err}(k+1) \\ \varepsilon_{err}(k+1) \end{bmatrix} = \begin{bmatrix} A_d & 0 \\ -C_d A_d & 1 \end{bmatrix} \begin{bmatrix} x_{err}(k) \\ \varepsilon_{err}(k) \end{bmatrix} + \begin{bmatrix} B_d \\ -C_d B_d \end{bmatrix} \begin{bmatrix} -K & g \end{bmatrix} \begin{bmatrix} x_{err}(k) \\ \varepsilon_{err}(k) \end{bmatrix}$$

然后得到如下控制系统的状态矩阵

$$\boldsymbol{A}_1 = \begin{bmatrix} A_d & 0 \\ -C_d A_d & 1 \end{bmatrix} \quad B_1 = \begin{bmatrix} B_d \\ -C_d B_d \end{bmatrix} \quad C_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

具有积分的状态反馈矢量为：

$$\boldsymbol{K}_1 = \begin{bmatrix} -K & g \end{bmatrix}$$

通过使用文件 oven2.m 得到矢量 $\boldsymbol{K}_1$。

oven2.m file

```
% Oven Control by state return with integration
```

```
Rm = 3; Rf = 0.1; Ca = 5000;
Cm = 10; k2 = 0.1; k1 = 100;
Te = 1;
b0 = k1*k2*Rf/(Rf*Rm*Cm*Ca);
a0 = 1/(Rf*Rm*Cm*Ca); a1 = 1/(Rm*Cm);

% Representation in the state space
A = [0 1 ; -a0 -a1];
B = [0 ;b0 ];
C = [1 0];
D = 0;
% Discretisation of the state representation
sys = ss(A, B, C, D);
sysd = c2d(sys, Te, 'zoh');
[Ad, Bd, Cd, Dd, Ts, Td] = ssdata(sysd);
A1 = [Ad zeros(2, 1);-Cd*Ad 1];
B1 = [-Bd; Cd*Bd];
C1 = [1 0 0]; D1 = 0;
```

对这个控制系统，选择二阶动力系统，固有频率ω_n=0.005 rad/s，阻尼系数为 0.9，以避免超调。

```
% Process dynamics
m = 0.9; wn = 1/200;
p1 = -2*exp(-m*wn*Te)*cos(wn*Te*sqrt(1-m^2));
p2 = exp(-2*m*wn*Te);

% Ackermann method
p = [1 p1 p2 0];
comb = ctrb(A1, B1);
disp(['rang = 'num2str(rank(comb))])

rang=3

K1 = acker(A1, B1, roots(p))

K1=
1. 0e+004*
-0.0136   -1.4819   0.0000

% Poles placement
[K2, prec, message] = place(A1, B1, roots(pa))

K2=
1. 0e+004*
-0.0136   -1.4819   0.0000
```

下面的 SIMULINK 模型用来检查控制系统响应（见图 4-17），文件 signal_oven4.m 用来显示结果。

伴随降低的线性指令，施加一个相当于 100℃的阶跃指令。

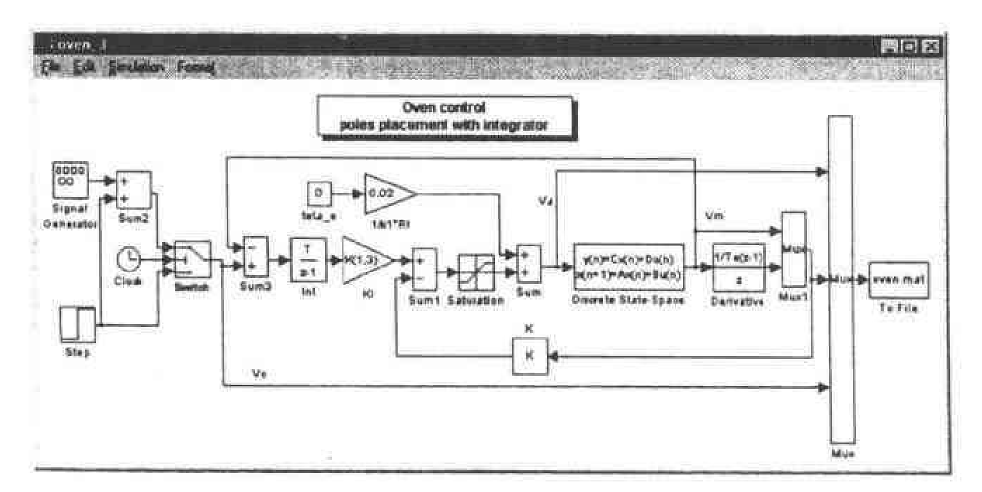


图 4-17　具有积分器的极点分布烤箱控制图

signal_oven4.m file

```
load oven.mat
t = signal(1, :);
Qa = signal(2, :);
Vm = signal(3, :);
dVm = signal(4, :);
Vc = signal(5, :);

figure(1)
plot(t, Vc)
hold on
plot(t, Vm, 'r'), hold off
title(' Vc  order and Vm measure ')
xlabel('Time')
ylabel('Voltage in V')
gtext('Vc\rightarrow')
gtext('\leftarrowVm')
figure(2)
plot(t, 100*Qa)
title('Heat quantity')
xlabel('Time'), ylabel('Power in W')
```

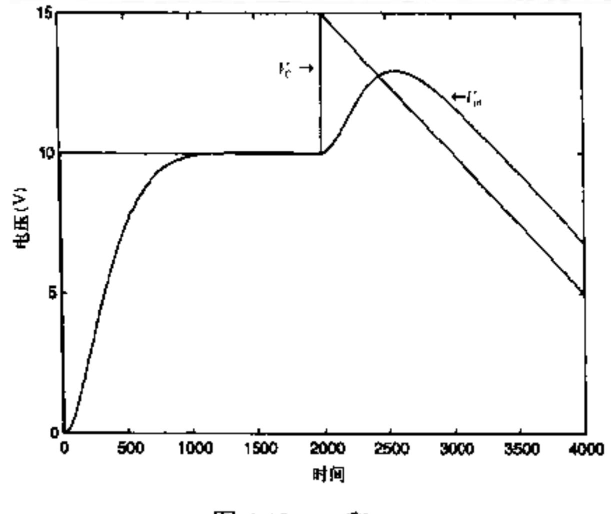


图 4-18　V_c和V_m

阶跃响应因积分作用而在没有任何超调或位置误差的情况下获得。另外，坡指令导致与轮廓相符的误差出现。为了消除它，有必要在前向通道中加入第 2 个积分。

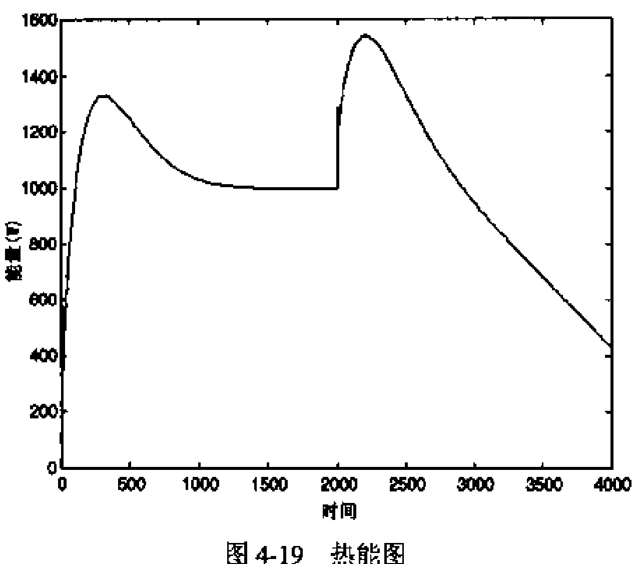


图 4-19　热能图

4.5　使用卡尔曼重构

假定不知道确切的过程模型以及温度 θ_m 的测量被中间的高斯白噪声破坏，在这个情况下，在卡尔曼随机重构的帮助下重建过程状态是可取的。

“控制系统工具箱”的卡尔曼函数可从 LTI 过程描述、Q 过程噪声变量和测量声音变量 R 中得到观测器的增益矢量。对于应用，假定

$$\begin{cases} E\left[v \quad v^{\mathrm{T}}\right] = R = 0.1 \\ E\left[\omega \quad \omega^{\mathrm{T}}\right] = Q = 0.1 \end{cases}$$

下面的命令可得到前面所述的离散过程描述的随机重构的增益矢量。

```
Q=0.1;
R=0.1;
[Kest,L,P]= kalman(sysd,Q,R)

L=
1.0e-003*
0.8183
0.0003
```

下面的 SIMULINK 模型代表了前面的修正过程，其中增加了卡尔曼重构（见图 4-20）。这样重构的状态矢量用来产生通过矢量 K 的状态反馈和用于通过矢量 $G_m[1 \quad 0]$的控制系统反馈的电压 V_m 图。加入的 $v(t)$和 $w(t)$噪声分别代表模型噪声和测量噪声。

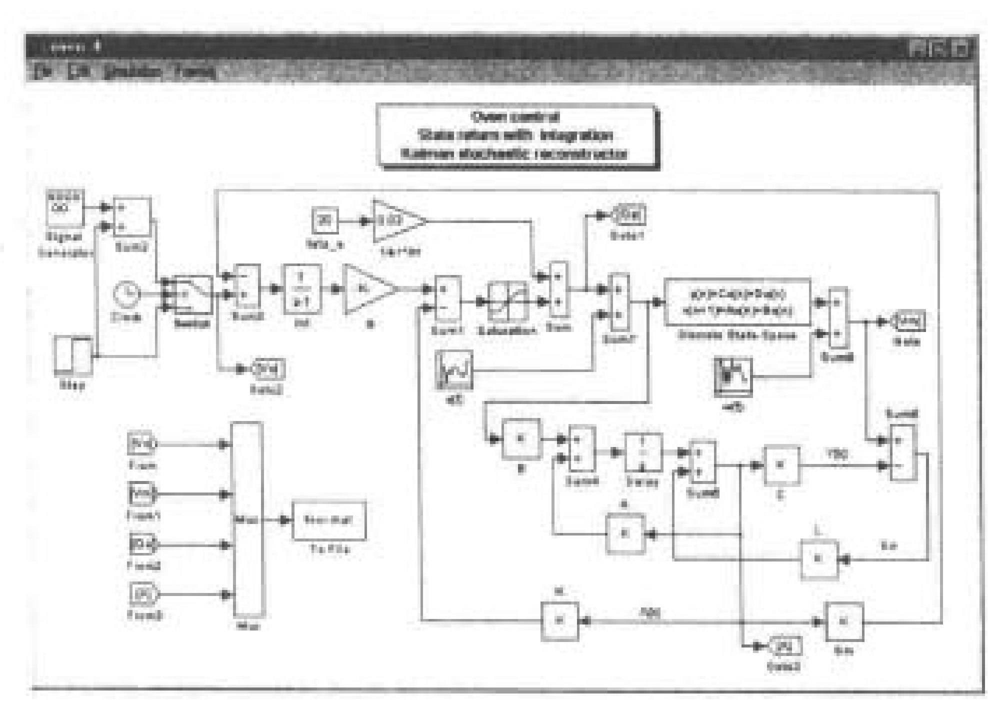


图 4-20　具有积分反馈卡尔曼随机重构的烤箱控制

文件 signal_oven5.m 代表在如前段中所述的指令下的模拟结果。

signal_oven5.m file

```
% Oven control

load oven.mat
t = signal(1, :);
Vc = signal(2, :);
Vm = signal(3, :);
Qa = signal(4, :);
x1 = signal(5, :);
x2 = signal(6, :);

figure(1)
plot(t, Vc)
hold on
plot(t, Vm, 'r')
hold off
title('Vc order and Vm measure ')
xlabel('Time')
ylabel('Voltage in V')
gtext('Vc\rightarrow')
gtext('\leftarrowVm')

figure(2)
plot(t, 100*Qa)
title('Heat quantity ')
xlabel('Time')
ylabel('Power in W')
```

随机重构效率通过比较期望变量、去除在 V_m 测量信号中高污染的模型噪声和测量噪声而清楚地呈现出来。

```
figure(3)
hx1 = line(t, x1)
xlabel('Time in seconds')
ylabel('Variable x1(t) = Vm(t)')
axet = axes('Position', get(gca,'Position'),'XAxisLocation','bottom',...
    'YAxisLocation','right','Color','none',...
    'XColor','k','YColor','k');
hx2 = line(t,x2,'color','r','parent',axet)
ylabel('Variable x2(t) = dVm(t)')
title(' x(t) Vector rebuilds ')
gtext('Vm(t)\rightarrow')
gtext('\leftarrow dVm(t)')
```

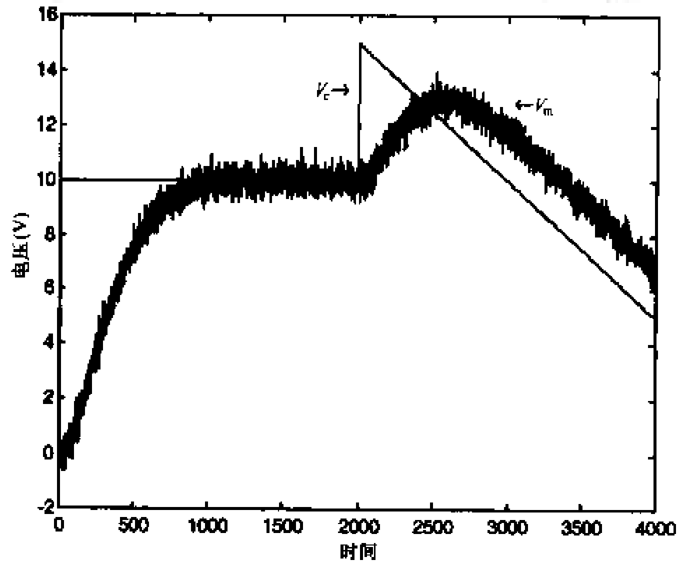


图 4-21 V_c和 V_m

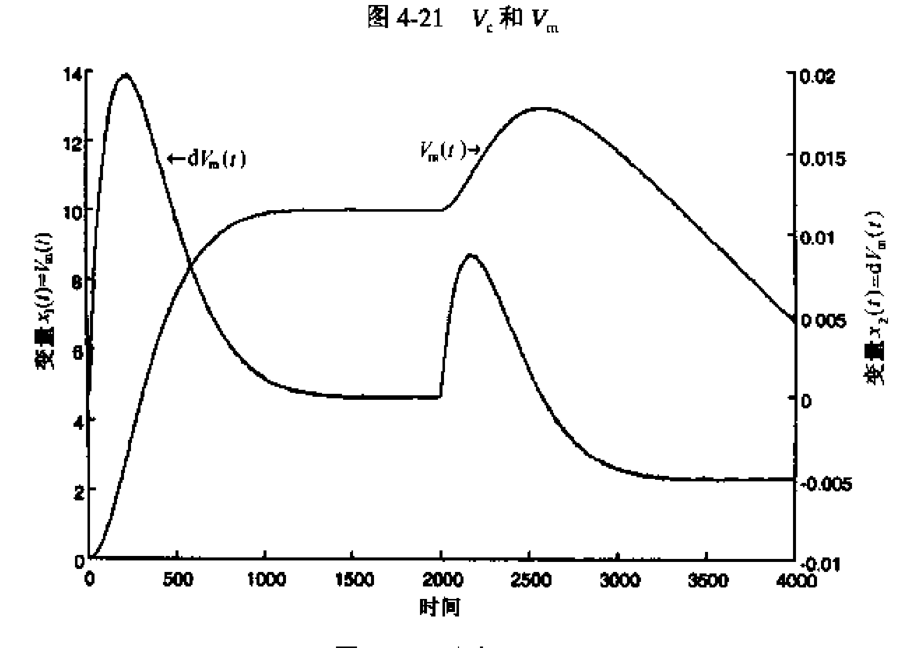


图 4-22 重建矢量 $x(t)$

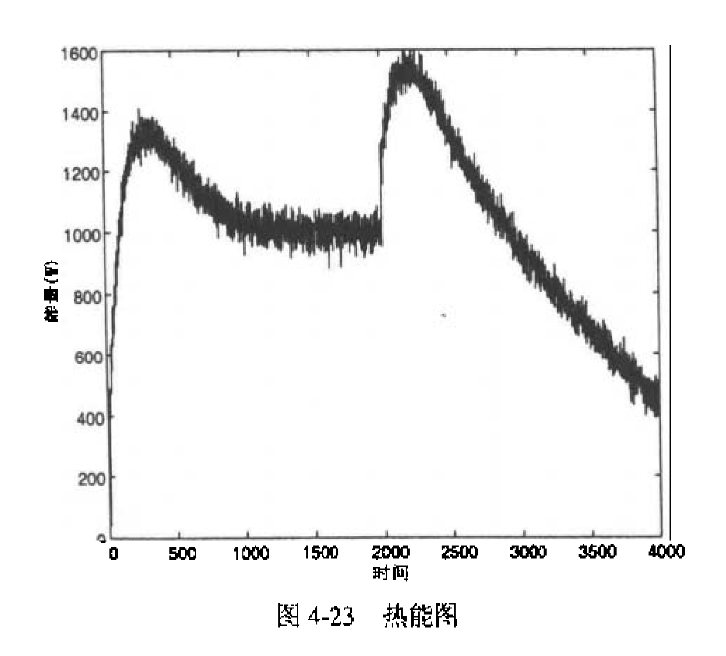


图 4-23　热能图

4.6　LQ 二次线性控制

现在考虑将状态返回矢量 K 的计算建于二次标准最小化的基础上。"控制系统工具箱"中的 lqr 和 dlqr 控制得到分别用于连续和离散过程的函数。控制 dlqr 返回修正过程极点、Riccati 方程的解答、从这个计算得出的最佳增益矢量 K 以及状态反馈方程：

$$u(k) = -Kx(k)$$

最小化如下二次指标：

$$J(u) = \sum_{k=1}^{\infty}\left[x(k)^{\mathrm{T}} Qx(k) + u(k)^{\mathrm{T}} Ru(k) + 2x(k)^{\mathrm{T}} Nu(k)\right]$$

这个指标出现 3 个调节术语：Q、R 和 N，默认 N=0。为了简化调节过程，应让 N=0，Q=I，I 是数(n,n)维的单位矩阵，n 是状态矢量中状态变量的数目。简化的指标变为

$$J(u) = \sum_{k=1}^{\infty}\left(x(k)^{\mathrm{T}} x(k) + u(k)^{\mathrm{T}} Ru(k)\right)$$

这个包含控制但不包含变量的指标不能得到包含在控制规则下的积分。因此，应保留在直链中包括积分的前面所述的控制函数图。用于 K 计算的状态式成为修正系统的状态式，也就是

$$\boldsymbol{A}_1 = \begin{bmatrix} A_d & 0 \\ -C_d A_d & 1 \end{bmatrix} \quad \boldsymbol{B}_1 = \begin{bmatrix} B_d \\ -C_d B_d \end{bmatrix} \quad \boldsymbol{C}_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

具有积分的状态反馈矢量为

$$\boldsymbol{K}_1 = \begin{bmatrix} -K & g \end{bmatrix}$$

文件 oven3.m 计算具有调节参数 R=2 000 的矢量 $\boldsymbol{K}$。

oven3.m file

```
% Oven control
% Quadratic linear command
Rm = 3; Rf = 0.1; Ca = 5000;
Cm = 10; k2 = 0.1; k1 = 100; Te = 1;
```

```
b0 = k1*k2*Rf/(Rf*Rm*Cm*Ca);
a0 = 1/(Rf*Rm*Cm*Ca); a1 = 1/(Rm*Cm);
% Representation in the state space
A = [0 1 ; -a0 -a1]; B = [0 ; b0 ];
C = [1 0]; D = 0;
% Discretisation of the state representation
sys = ss(A, B, C, D);
sysd = c2d(sys, Te, 'zoh');
[Ad, Bd, Cd, Dd, Ts, Td] = ssdata(sysd);
A1 = [Ad zeros(2, 1);-Cd*Ad 1]; B1 = [-Bd;Cd*Bd];
C1 = [1 0 0]; D1 = 0;
% Kalman estimator
Q = 0.1; R = 0.1;
[Kest, L, P] = kalman(sysd, Q, R);
% lqr command
Q1 = 1*eye(3); R1 = 2000;
[K, S, e] = dlqr(A1, B1, Q1, R1)

K=-4.3258  -116.5345   0.0223

S=
1.0e+009*
0.0049   0.1301   0.0000
0.1301    3.5028   -0.0007
-0.0007   0.0000

e=
0.9692
0.9951+0.0048i
0.9951-0.0048i
```

前面图形中所述的具有计算出的矢量 K 的 oven_3.mdl SIMULINK 模型的仿真给出如下结果。

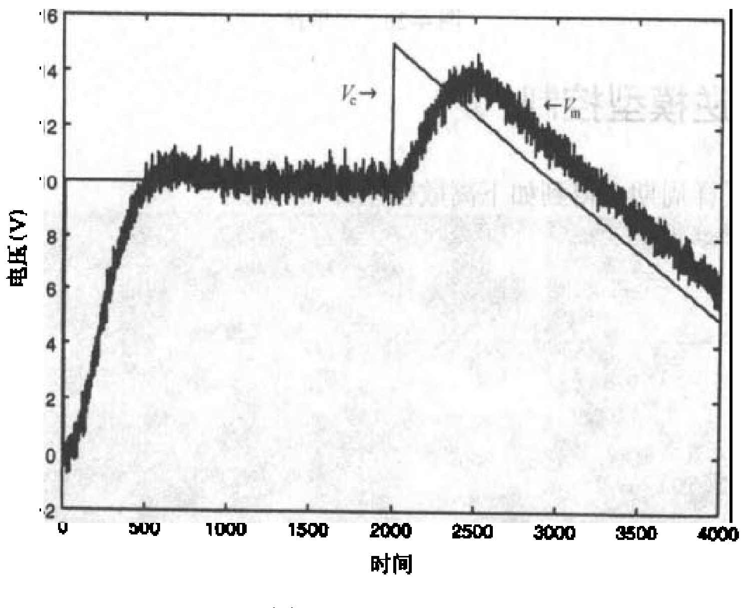


图 4-24　V_c 和 V_m

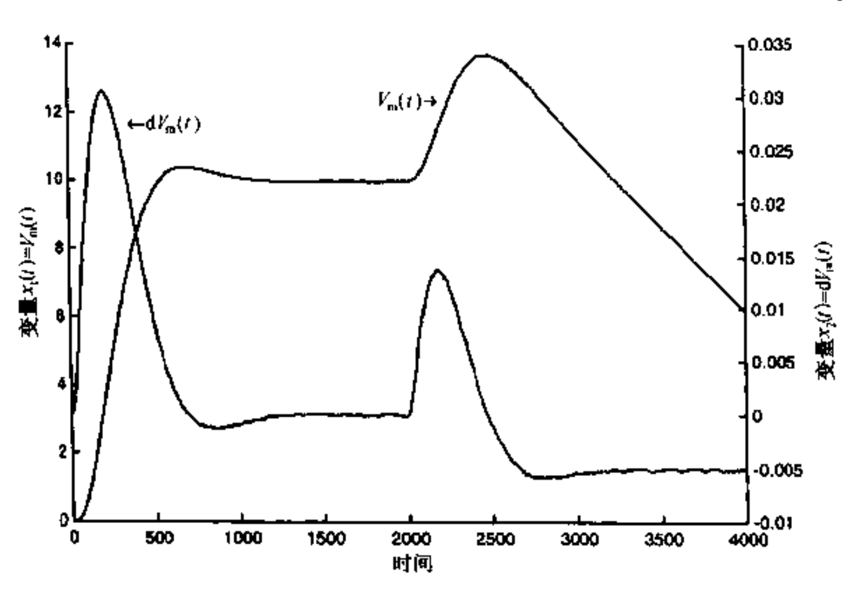


图 4-25　重构矢量 $x(t)$

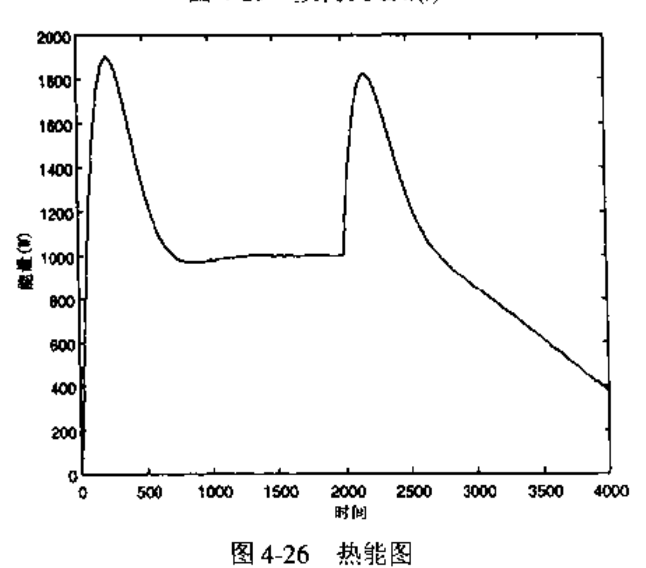
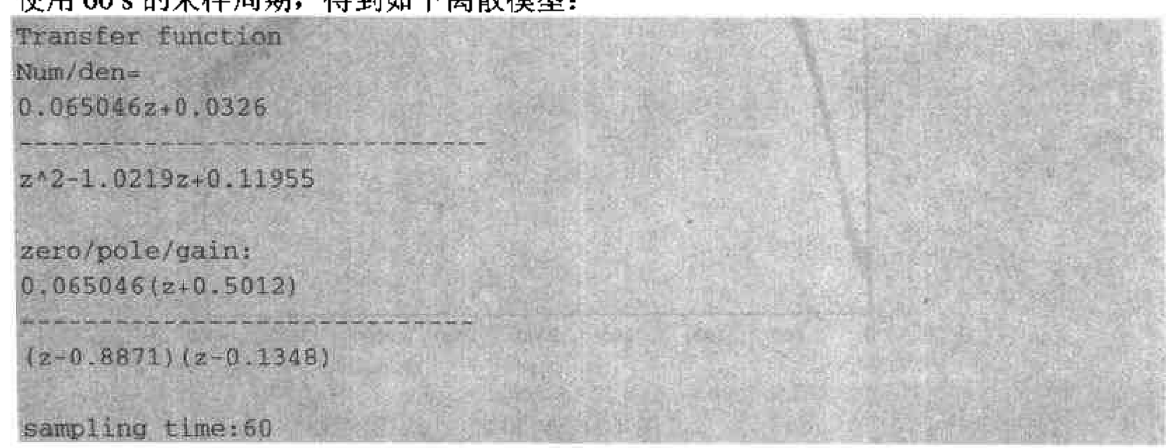


图 4-26　热能图

4.7　神经元逆模型控制

使用 60 s 的采样周期，得到如下离散模型：



```
Transfer function
Num/den=
0.065046z+0.0326
-------------------------------
z^2-1.0219z+0.11955

zero/pole/gain:
0.065046(z+0.5012)
------------------------------
(z-0.8871)(z-0.1348)

sampling time:60
```

建议通过逆模型实现烤箱神经元控制。在实际的烤箱中，也许在工作区域没有线性化。为得到一个在任何工作点都能适用的神经元模型，给这个过程施加一个带有伪随机二进制序列的斜坡信号。因此写出用于神经网络外部学习的输入/输出文件。这由文件 orderl.m 和 nr_signal.mdl 的 SIMULINK 模型产生。

order1.m file

```
% generation of a rise signal + prbs
N = 1000; brps = (-1).^(1:10);
for i = 11:N
  brps(i) = -brps(i-10)*brps(i-7);
end
rise = 50*(0:N-1)/N; u = rise+5*brps;
u = u(100:900); t = 1:length(u);
```

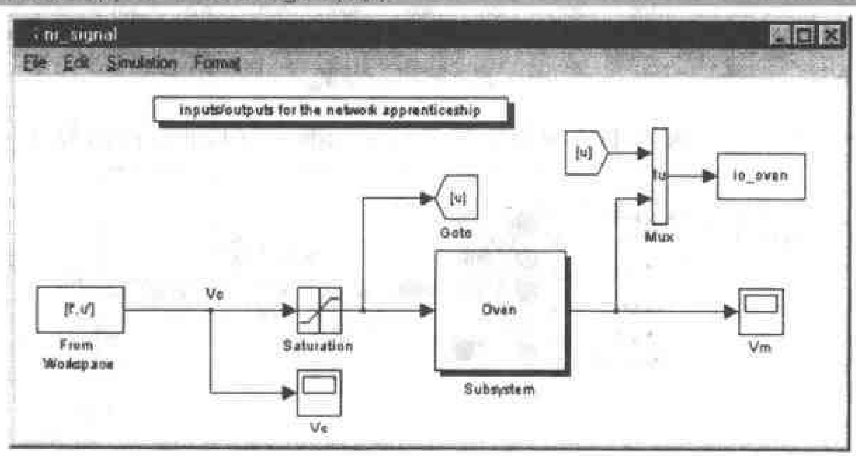


图 4-27 网络训练的输入输出

◆ 信号显示

```
" load io_oven
" u=uy(2,: ); y=uy(3, : );
" figure(1), stairs(u), title('Control signal ')
" xlabel('samples'),grid
" figure(2), plot(y),title('Vm output signal')
xlabel('samples'), grid
```

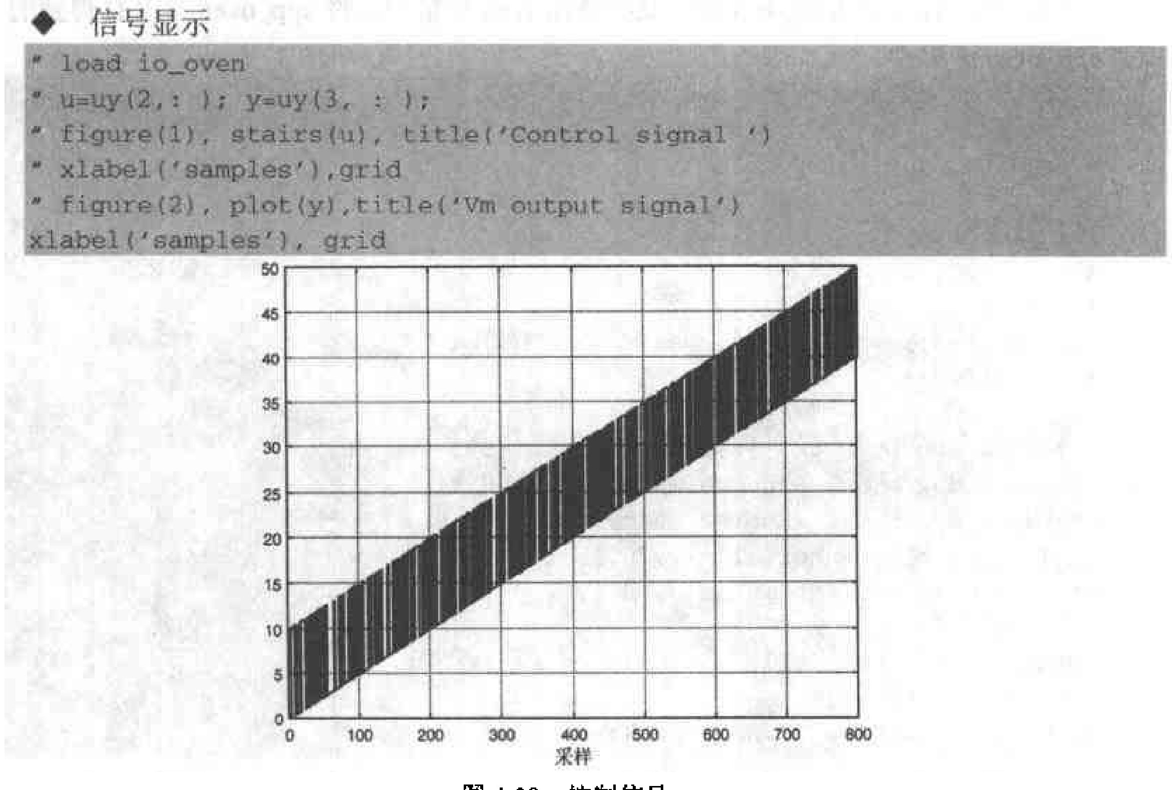


图 4-28 控制信号

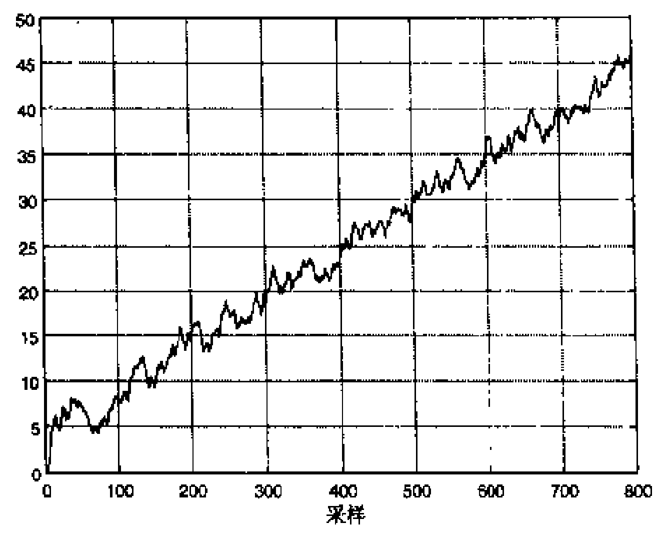


图 4-29　输出信号 V_m

选择如下结构，这里 N 和 DN 块分别代表信号标准运算和非标准运算（见图 4-30）。

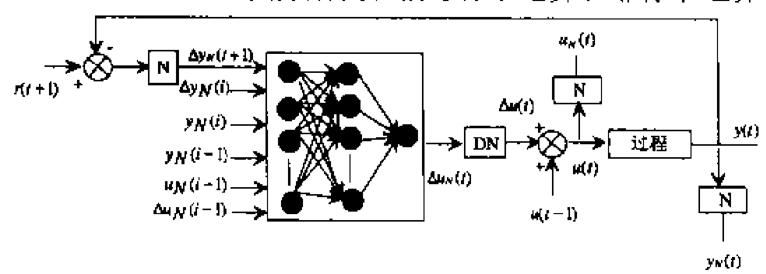


图 4-30　结构图

网络学习是借助于从 io_oven.mat 文件读出的输入/输出文件 app_oven.m 文件得到的。

app_oven.m file

```
% Learning of the oven inverse model

% reading of the I/O file

load io_oven
u = uy(2, :);
y = uy(3, :);
du = diff(u);
dy = diff(y);

% de, du and dy normalization between -0.9 and +0.9
[du_N, a_du, b_du] = normalis(du, 0.1, 0.9);
[dy_N, a_dy, b_dy] = normalis(dy, -0.9, 0.9);
[y_N, a_y, b_y] = normalis(y, 0.1, 0.9);
[u_N, a_u, b_u] = normalis(u, 0.1, 0.9);

% Number of input cells (6), hidden cells (7),
% and of output (1)
Nb_Iu = 6; Nb_Hu = 7; Nb_Ou = 1;

% Random initialization of the W and Z weights, and of w0 and
```

```
% z0 bias
W = rand(Nb_Hu, Nb_Iu)-0.5;
w0 = rand(Nb_Hu, 1)-0.5;

Z = rand(Nb_Ou, Nb_Hu)-0.5;
z0 = rand(Nb_Ou, 1)-0.5;

N_iter = 1000; % iterations number
eta = 0.8;     % learning gain
k = 0;
while k<=N_iter
  k = k+1;
  for i = 3:length(u)-2
   % stimulus number i
   x=[dy_N(i+1) dy_N(i) y_N(i) y_N(i-1) u_N(i-1) du_N(i-1)]';

   % responses of the hidden layer cells
   h = logsig(W*x+w0);

   % cells responses of the output layer cells
   o(i) = logsig(Z*h+z0);

   % error in output of the network
   e(i) = du_N(i)-o(i);
   % error to back-propagate
   delta(i) = o(i).*(ones(Nb_Ou, 1)-o(i)).*e(i);

   % updating of matrixes of Z weight and z0 bias
   Z = Z+eta*delta(i)*h';
   z0 = z0+eta*delta(i);

   % error in hidden layer
   dh = h.*(ones(size(h))-h).*(Z'*delta(i));

   % updating of matrixes of W weight and w0 bias
   W = W+eta*dh*x';
   w0 = w0+eta*dh;
 end %  for loop

 Weights_W(:,:,k) = W;
 Weights_Z(:,:,k) = Z;
 bias_w0(:,:,k) = w0;
 bias_z0(:,:,k) = z0;

 % saving of weights and bias in the weights1.mat file
 save wz_oven Weights_W Weights_Z bias_w0 bias_z0

 home
 k
end % while loop
```

在 1 000 次迭代后，得到如下的权重和偏离值。

```
W=
-0.5201   -0.1617   0.2582  -0.0249   0.8535   0.4976
-1.2405    0.2536  -0.2031   0.5819   0.3654   0.2795
-0.2795    0.0564   0.0507   0.1885   0.6913   0.2240
-0.7950   -1.5213   0.5586  -0.8173  -0.2243  -1.5727
-1.0651   -2.2353   1.3913  -1.0110  -1.1510  -2.6391
-4.5552    0.7292  -1.9806   0.6539   0.5970   1.0865
 0.6882   -0.3961   1.4708  -0.4122   0.0300  -0.5119
w0=
0.5177
-0.3885
0.2072
0.9595
1.4417
-0.4672
0.5627
z=
   -0.5105  -0.9478  -0.3894   1.3862   3.2163  -3.7336   1.4134
z0=
   -0.2664
```

不同的权重和偏离值的演变在如下曲线中表示出来（见图 4-31 至图 4-34)。

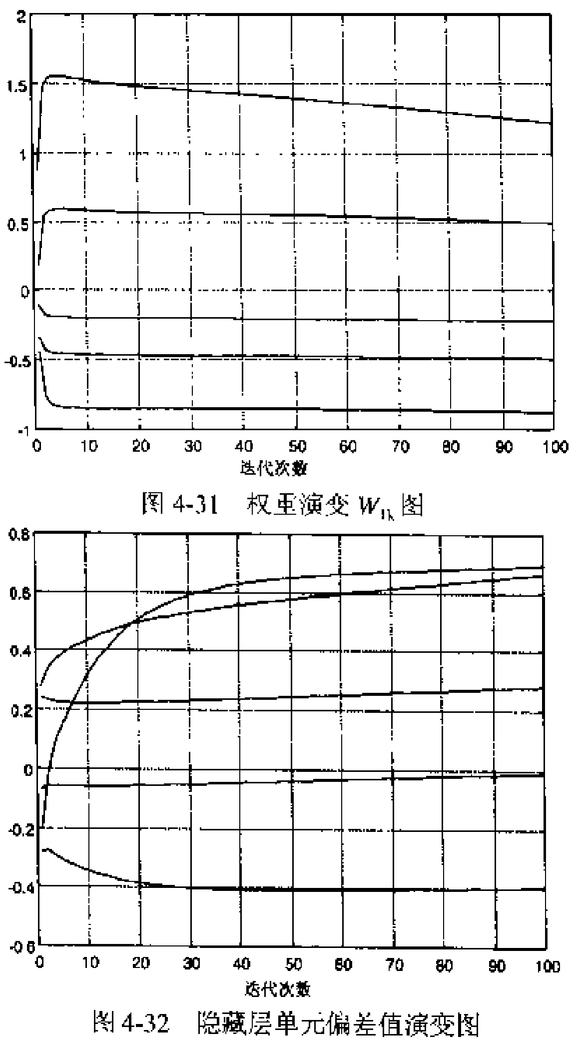


图 4-31　权重演变 W_{1k} 图

图 4-32　隐藏层单元偏差值演变图

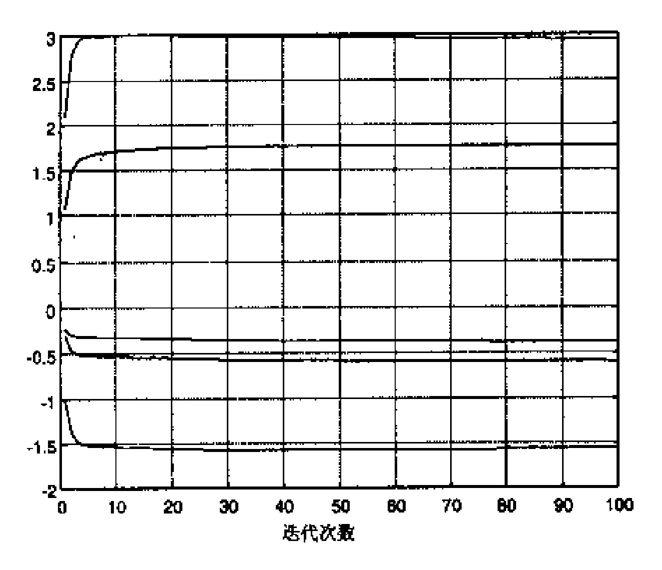


图 4-33　权重 Z 演变图

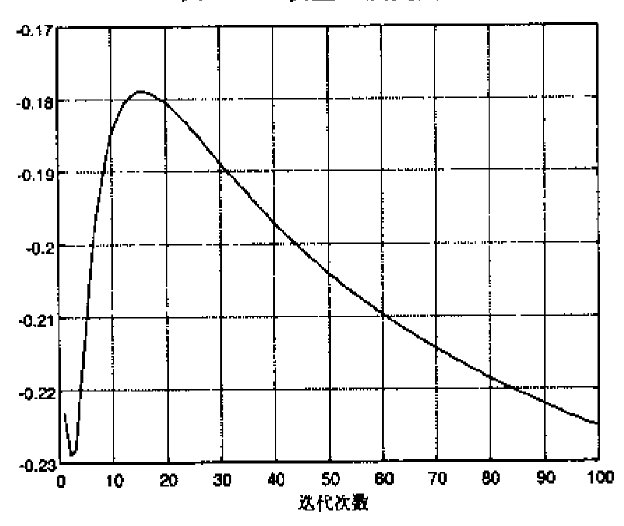


图 4-34　输出单元偏离演变图

为检查学习质量，下面给出文件 io_oven.mat 读出的实际控制变量，这是在网络输出非归一化前得到的（见图 4-35）。

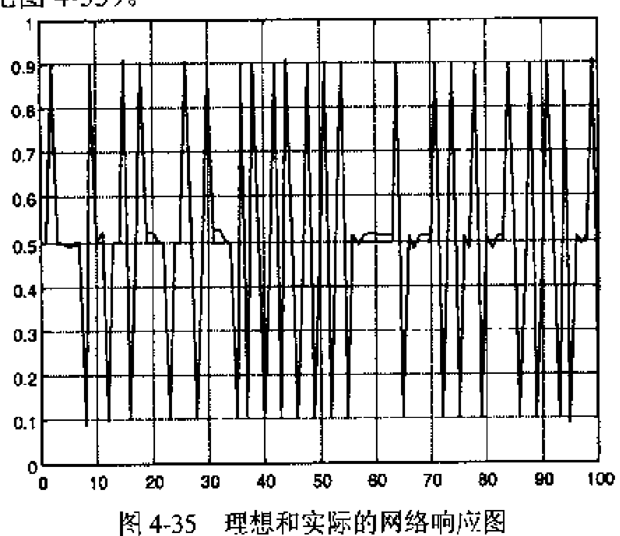


图 4-35　理想和实际的网络响应图

网络的实际和希望所得的输出误差绝对值小于 0.01（见图 4-36）。

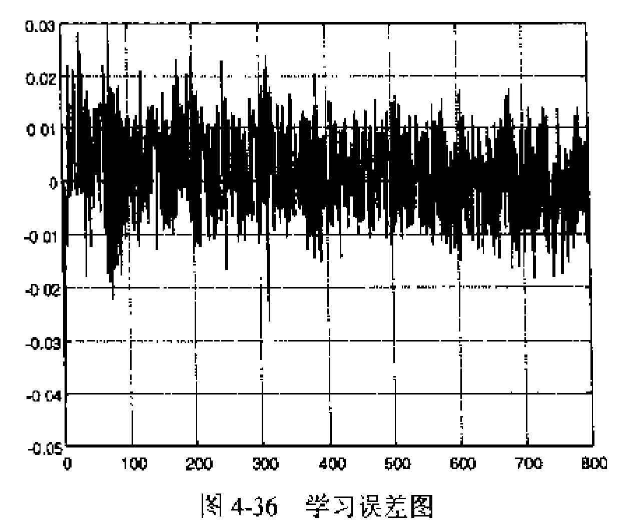


图 4-36　学习误差图

神经控制概要由如下 SIMULINK 模型表示（见图 4-37），这里对应于正常控制的网络输出通过增益 k 而增加了。

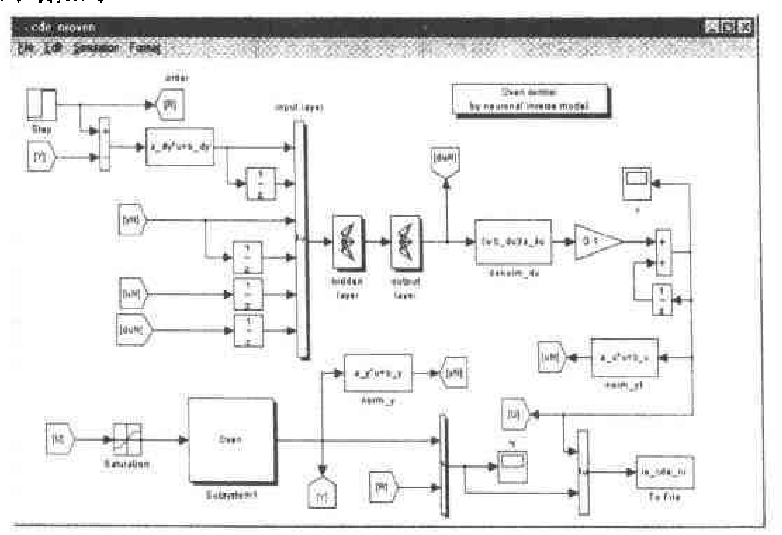


图 4-37　神经逆模型的烤箱控制

在增益 k=0.1 下，在如下同样的图中表示控制、输出和指令信号。因为积分控制为网络提供了控制变量，所以位置误差为零。响应时间可通过增益值 k 来调节。

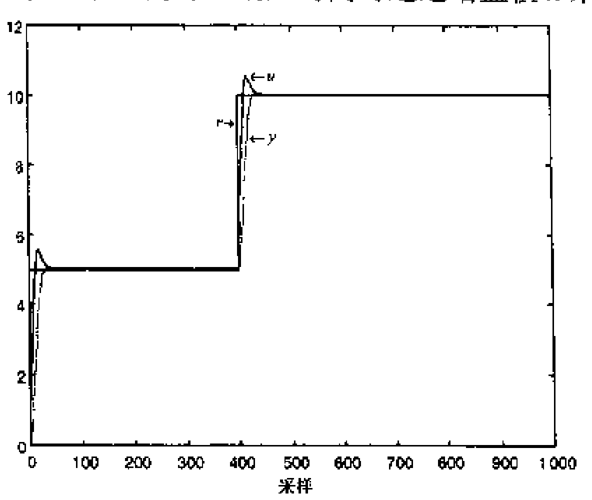


图 4-38　烤箱神经控制（增益 k=0.1）

如果很小地增加 k 值（k=0.3），响应时间减少，但在稳定状态下出现振荡。

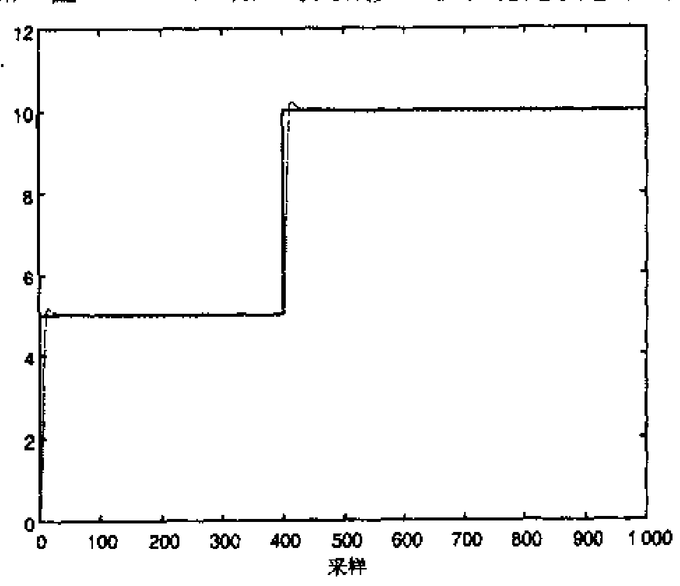


图 4-39　指令和输出信号（增益 k=0.3）

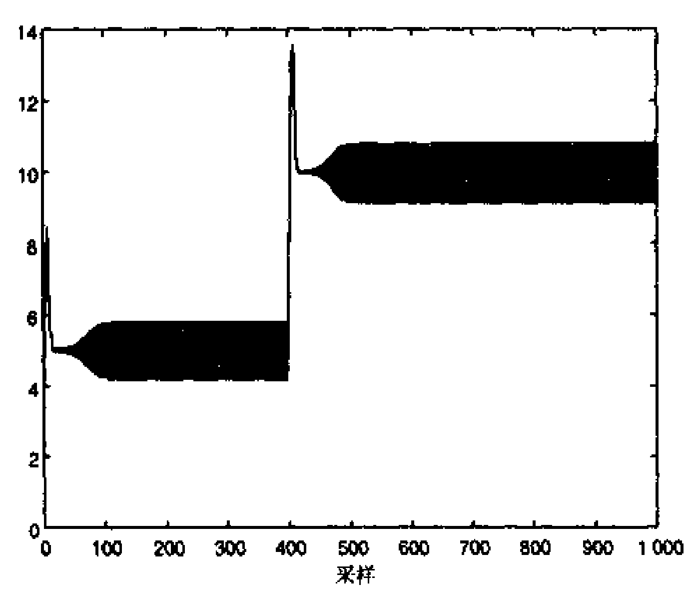


图 4-40　烤箱神经控制（增益 k=0.3）

应用 5　具有悬挂物的移动高架吊车

为了调整在力 F 作用下沿 x 轴移动的质量为 M 的高架吊车和挂在长缆绳上的块 m 的自由度，必须建立一个模型。块 m 与垂直点成角 θ 振荡（见图 5-1）。

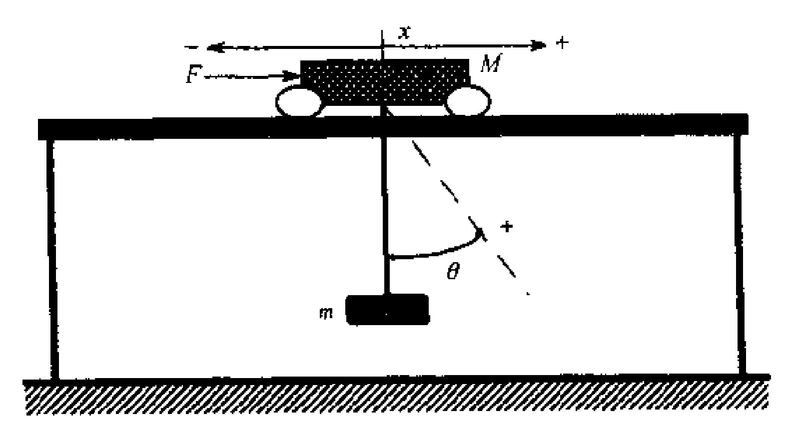


图 5-1　高架吊车模型图

5.1　具有 2 个自由度的移动高架吊车模型

为此，使用如下方程定义的拉格朗日方程：

$$\begin{cases} L = E_c - E_p \\ \dfrac{d}{dt}\left(\dfrac{\partial L}{\partial q'}\right) - \dfrac{\partial L}{\partial q} + \dfrac{\partial D}{\partial q'} = F_q \end{cases}$$

其中：

- q　$x(t)$和 $\theta(t)$的自由度；
- D　由于摩擦而消耗的能量；
- F_q　由自由度 q 产生的力；
- E_c和 E_p　系统的动能和势能。

5.1.1　系统移动时的动能

$$E_c = \frac{1}{2}M\left(\frac{dx}{dt}\right)^2 + \frac{1}{2}mV_m^{\ 2} = \frac{1}{2}M\dot{x}^2 + \frac{1}{2}mV_m^{\ 2}$$

组件速度由下式决定：

$$V_m^{\ 2} = \dot{x}^2 + \left(l\dot{\theta}\right)^2 + l\dot{x}\dot{\theta}\cos\theta$$

于是

$$E_c = \frac{1}{2}(M+m)\dot{x}^2 + \frac{1}{2}ml^2\dot{\theta}^2 + ml\dot{x}\dot{\theta}\cos\theta$$

5.1.2 系统的势能

$$E_p = mgl - mgl\cos\theta$$

5.1.3 在 $q(t) = \theta(t)$ 自由度下的拉格朗日方程

如果 D=0，则自由度相等，因此不考虑总摩擦损失

$$L = \frac{1}{2}(M+m)\dot{x}^2 + \frac{1}{2}ml^2\dot{\theta}^2 + ml\dot{x}\dot{\theta}\cos\theta - mgl(1-\cos\theta)$$

根据自由度 θ 所施加的力为零

$$F_\theta = 0$$

于是

$$\frac{\partial L}{\partial \dot{\theta}} = ml^2\dot{\theta} + ml\dot{x}\cos\theta$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = ml^2\ddot{\theta} + ml\ddot{x}\cos\theta - ml\dot{x}\dot{\theta}\sin\theta$$

$$\frac{\partial L}{\partial \theta} = -ml\dot{x}\dot{\theta}\sin\theta - mgl\sin\theta$$

简化后，给出第一拉格朗日方程：

$$l\ddot{\theta} + \ddot{x}\cos\theta + g\sin\theta = 0$$

5.1.4 在 $q(t) = x(t)$ 自由度下的拉格朗日方程

根据自由度 x 给吊车施力：

$$F_x = F(t)$$

$$\frac{\partial L}{\partial \dot{x}} = (M+m)\dot{x} + ml\dot{\theta}\cos\theta$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{x}}\right) = (M+m)\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta$$

$$\frac{\partial L}{\partial x} = 0$$

简化后，给出拉格朗日第二方程：

$$(M+m)\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F(t)$$

5.1.5 操作点附近的线性模型

得到的模型是非线性的，且不同自由度下有不同的方程。如果只考虑在操作点 θ_0=0 附近只有很小的 θ 变化，可考虑如下的简化：

$$\begin{cases}\cos\theta = 1 \\ \sin\theta = \theta\end{cases}$$

另外，有

$$\ddot{\theta}\cos\theta - \dot{\theta}^2\sin\theta = \frac{\mathrm{d}}{\mathrm{d}t}\left(\dot{\theta}\cos\theta\right)$$

这给出如下的线性微分方程：

$$\begin{cases}(M+m)\ddot{x}+ml\ddot{\theta}=F(t)\\ \ddot{x}+l\ddot{\theta}+g\theta=0\end{cases}$$

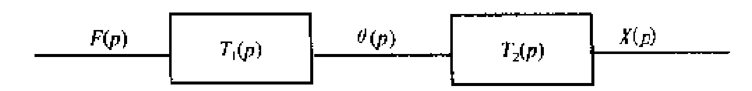
根据 $x(t)$和 $\theta(t)$的微分系统给出：

$$\begin{cases}\ddot{\theta}(t)=-g\dfrac{(M+m)}{Ml}\theta(t)-\dfrac{1}{Ml}F(t)\\ \ddot{x}(t)=g\dfrac{m}{M}\theta(t)+\dfrac{1}{M}F(t)\end{cases}$$

5.2 系统的传递函数

根据 $\theta(t)$写出的微分方程与力 $F(t)$相联系，然而，由于 $x(t)$的方程依赖于 $F(t)$和 $\theta(t)$，可由如下所示的串联传递函数来描述过程：

$$T_1(p)=\frac{\theta(p)}{F(p)}\qquad T_2(p)=\frac{X(p)}{\theta(p)}$$



微分系统的拉普拉斯变换如下：

$$p^2\theta(p)+g\frac{(M+m)}{Ml}\theta(p)=-\frac{1}{Ml}F(p)$$

$$p^2X(p)=g\frac{m}{M}\theta(p)+\frac{1}{M}F(p)$$

于是：

$$T_1(p)=\frac{\theta(p)}{F(p)}=\frac{-1}{Mlp^2+g(M+m)}$$

$$T_2(p)=\frac{X(p)}{\theta(p)}=-\frac{g+lp^2}{p^2}$$

5.2.1 开环过程的阶跃响应

在吊车上施加一个阶跃力 $F(t)=F_0u(t)$ 。忽略所有的摩擦，可预测悬挂物的持续振荡运动：

$$F(p)=\frac{F_0}{p}$$

$$\theta(p)=\frac{F_0}{p}\times\frac{-1}{Mlp^2+g(M+m)}=\frac{k_1}{p}+\frac{k_2p}{Mlp^2+g(M+m)}$$

其中：

$$\begin{cases} k_1 = \dfrac{-F_0}{(M+m)g} \\ k_2 = \dfrac{F_0 ml}{(M+m)g} \end{cases}$$

于是

$$\theta(p) = \frac{-F_0}{(M+m)g}\left[\frac{1}{p} - \frac{p}{p^2 + \dfrac{(M+m)g}{Ml}}\right]$$

这得到如下原型：

$$\theta(t) = \frac{-F_0}{(M+m)g}\left[1-\cos\left(\sqrt{\frac{(M+m)}{Ml}g}\right)t\right]u(t)$$

$$T_2(p) = \frac{X(p)}{F(p)} = \frac{g+lp^2}{p^2[Mlp^2+g(m+M)]} = \frac{1}{(M+m)p^2} + \frac{ml}{(M+m)} \times \frac{1}{[mlp^2+g(M+m)]}$$

总有

$$F(p) = \frac{F_0}{p}$$

于是

$$X(p) = \frac{F_0}{(M+m)p^3} + \frac{F_0 ml}{g(M+m)^2}\left(\frac{1}{p} - \frac{Mlp}{Mlp^2 + g(M+m)}\right)$$

原型可写为：

$$x(t) = \left[\frac{F_0}{2(M+m)}t^2 + \frac{F_0 ml}{g(M+m)^2}\left(1-\cos\sqrt{\frac{g(M+m)}{Ml}}t\right)\right]u(t)$$

具有如下数据：

M=10 kg，m=5 kg，l=1 m，g=9.81≈10，F_0=1。

得到 $\theta(t)$、$x(t)$的表达式：

$$\theta(t) = \frac{-1}{150}(1-\cos 3.873t)u(t)$$

$$x(t) = \frac{1}{30}t^2 + 0.002\,2(1-\cos 3.873t)u(t)$$

$x(t)$方程近似为：

$$x(t) = \frac{1}{30}t^2$$

很容易获得 1.62 s 周期的振荡运动，悬挂物对吊车的抛物线位移影响很小。

5.2.2 模型的建立与检测

◆ SIMULINK 模型（见图 5-2）

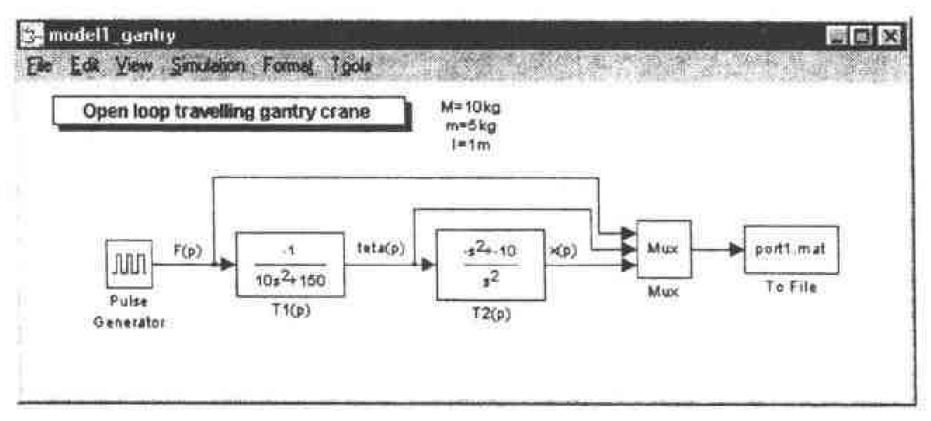


图 5-2　移动高架吊车开环控制图

◆　仿真结果

仿真结果保存在“Signals”变量名下的数据文件 port1.mat 中。为利用这些结果以及呈现控制 $F(t)$的响应，执行如下的 gantryl.m 文件。

gantry1.m file

```
% travelling gantry crane in open loop
% reading the simulation results data file
load port1.mat
% plotting the F(t) and Teta(t) curves
figure(1)
t = signals(1,:); F = signals(2,:); Teta = signals(3,:);
hf = line(t,F);
xlabel('time in seconds'), ylabel('Force in N')
axis([0 7 -0.1 1.1])
axet = axes('Position',get(gca,'Position'),...
    'XAxisLocation','bottom',...
    'YAxisLocation','right','Color','none',...
    'XColor','k','YColor','k');
ht = line(t,Teta,'color','r','parent',axet);
ylabel('angle evolution'), grid
title('F(t) control in N and \Theta(t) response in rd')
gtext('F(t)\rightarrow'), gtext('\leftarrow \theta(t)')
% Plotting F(t) and x(t) curves
x = signals(4,:); figure(2), plot(t,F), hold on
plot(t,x),hold off, grid
title('F(t) control and x(t) response')
ylabel('Open loop travelling gantry crane')
xlabel('time in seconds')
gtext('Force in N'), gtext('x position in meters')
```

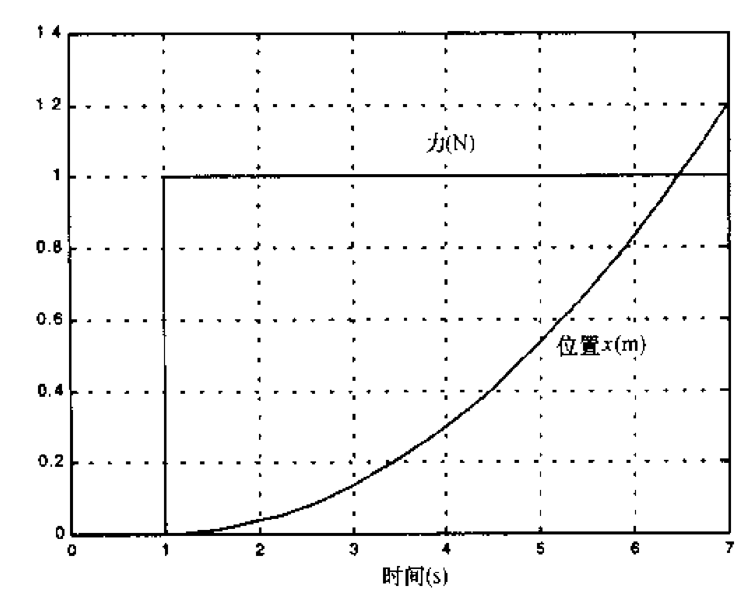


图 5-3　控制 $F(t)$和响应 $x(t)$

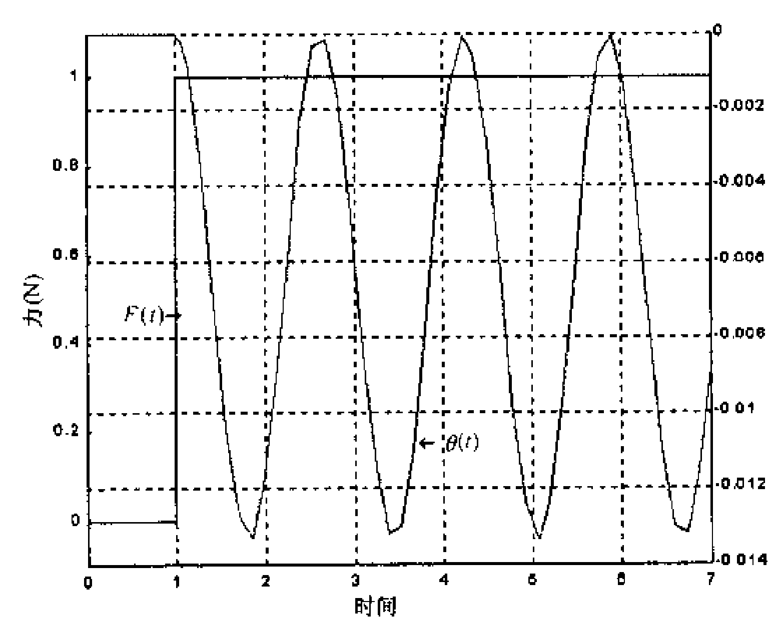


图 5-4　控制 $F(t)$和响应 $\theta(t)$

注意

为让吊车停止，最好在 $\theta(t)$ 经过零的瞬间取消力 $F(t)$。否则，悬挂物将继续振荡。可用一新的仿真试验来检验它。为此在块振荡的 2 个周期间在移动高架吊车上施加 1 N 的力。

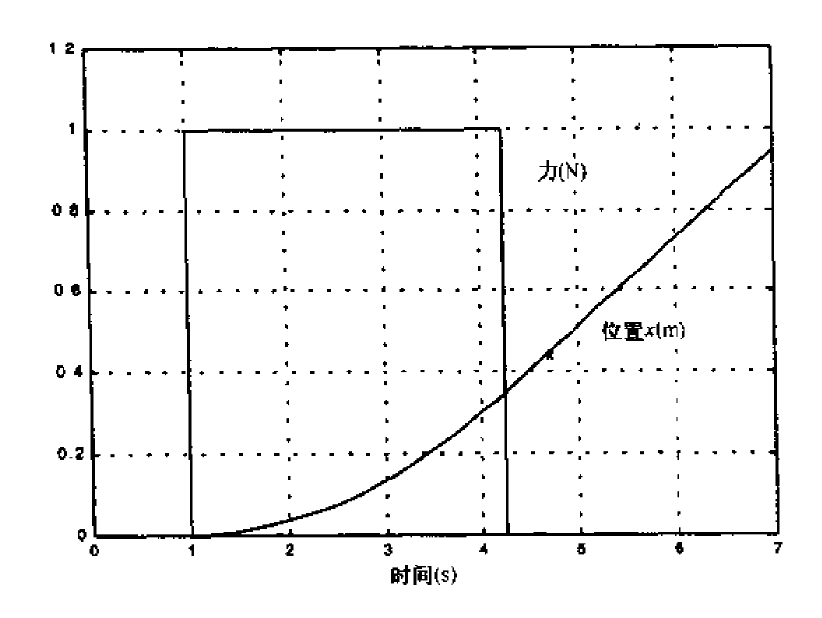


图 5-5 控制 $F(t)$和响应 $x(t)$

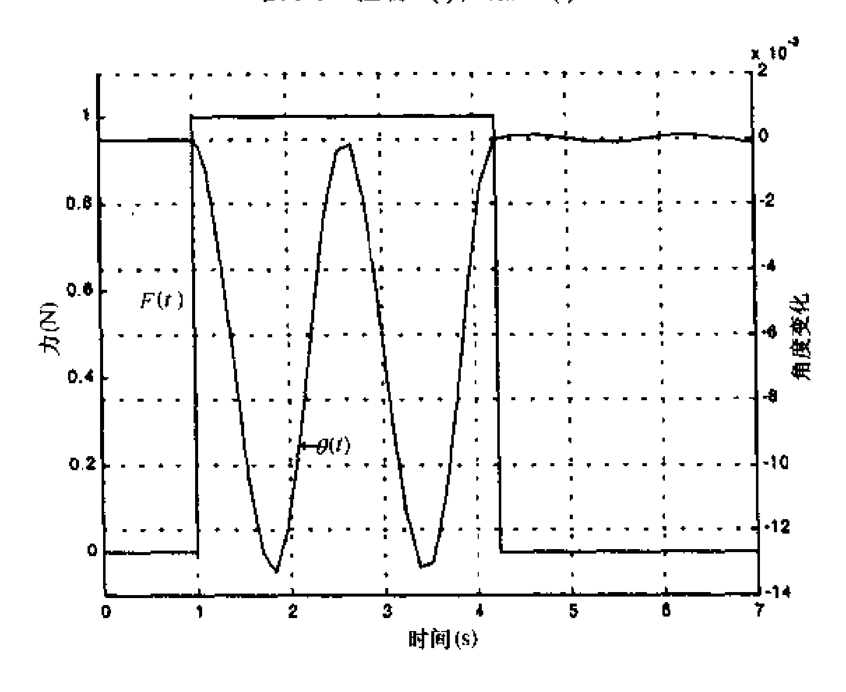


图 5-6 控制 $F(t)$和响应 $\theta(t)$

在 $\theta(t)$经过零点时取消力 $F(t)$，避免了残余振荡。

5.3 $\theta(t)$角位置的调节

首先调节 $\theta(t)$角位置来消除悬挂物的振荡。为此，执行一个具有微分反馈的闭环调节（见图 5-7）。

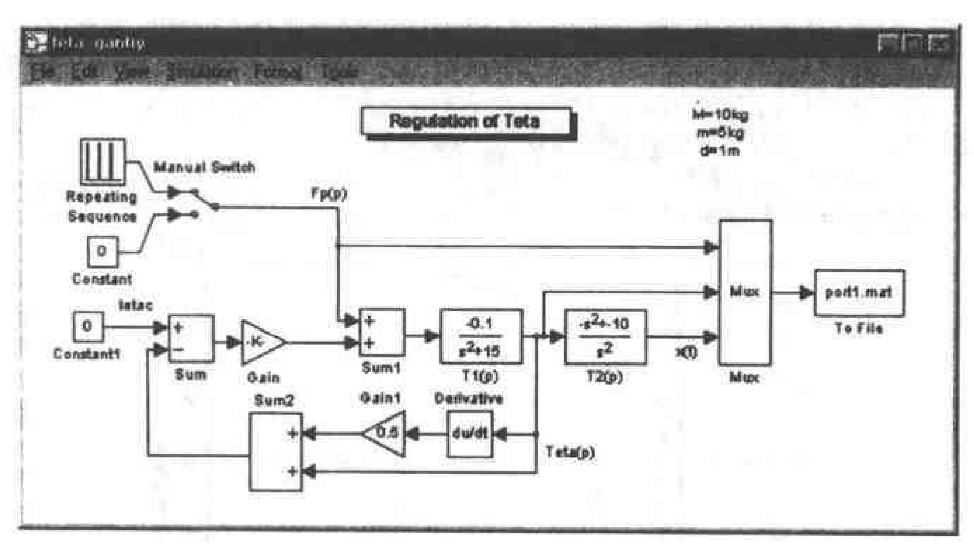


图 5-7 具有微分反馈的闭环调节图

$$T_1(p)=\frac{-1}{Mlp^2+(M+m)g}=\frac{b_0}{p^2+a_0}$$

$$\frac{\theta(p)}{F(p)}=\frac{T_1(p)}{1+K_1(1+K_2p)T_1(p)}=\frac{b_0}{a_0+K_1b_0}\times\frac{1}{1+\dfrac{K_1K_2b_0}{a_0+K_1b_0}p+\dfrac{1}{a_0+K_1b_0}p^2}$$

给出恰当的振动频率ω_n和阻尼系数ξ：

$$\begin{cases}\omega_n=\sqrt{a_0+K_1b_0}=\sqrt{15-0.1K_1}\\ \xi=\dfrac{1}{2}\times\dfrac{K_1K_2b_0}{\sqrt{a_0+K_1b_0}}=\dfrac{-0.05K_1K_2}{\sqrt{15-0.1K_1}}\end{cases}$$

在ω_n =10 rad/s 和ξ =0.7 时，给出 K_1 =−850、K_2 =0.165。

文件 gantry2.m 呈现的是在脉冲$f(t)$作用下的$x(t)$和 $\theta(t)$响应。

gantry2.m file

```
% gantry in closed loop without frictions
% Laplace's method
% Regulation of the angular position Teta
M = 10; m = 5; d = 1; g = 10; k1 = m*g/M;
k2 = -(M+m)*g/(M*d); k3 = 1/M; k4 = 1/(M*d);
load port1.mat
figure(1)
t = outputs(1,:); F = outputs(2,:); teta = outputs(3,:);
x = outputs(4,:);
hf = line(t,F);
xlabel('time in seconds'), ylabel('Force in N')
axis([0 3 -0.1 1.1])
axet = axes('Position',get(gca,'Position'),'XAxisLocation','bottom',...
     'YAxisLocation','right','Color','none',...
     'XColor','k','YColor','k');
ht = line(t,teta,'color','r','parent',axet);
ylabel('\theta(t) angle evolution'), grid
title('F(t) disturbance in N and \Theta(t) response in rd')
```

```
gtext('\leftarrow F(t) disturbance')
gtext('\leftarrow \theta(t) angle')
figure(2), plot(t,x), grid, title('x(t) position in m')
ylabel('Regulation of the position ')
xlabel('time in seconds')
gtext('x(t) truck position')
```

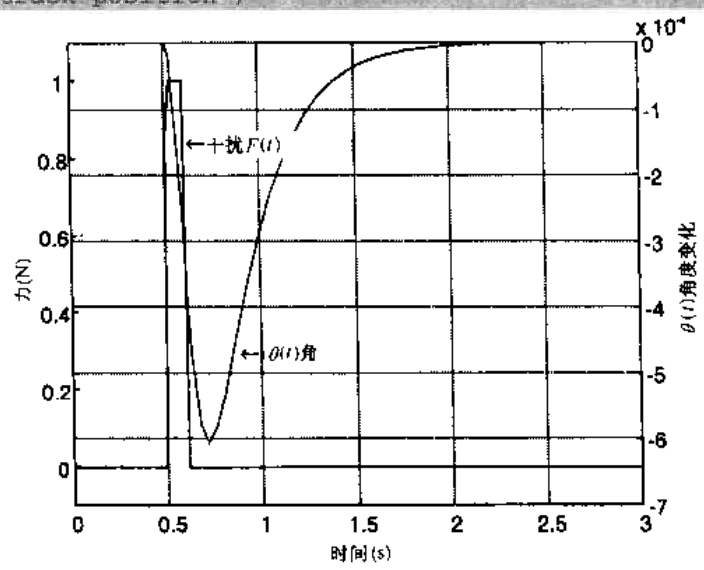


图 5-8 干扰 $F(t)$和响应$\theta(t)$

可以看到，θ 角的修正使响应小于 0.5 s。

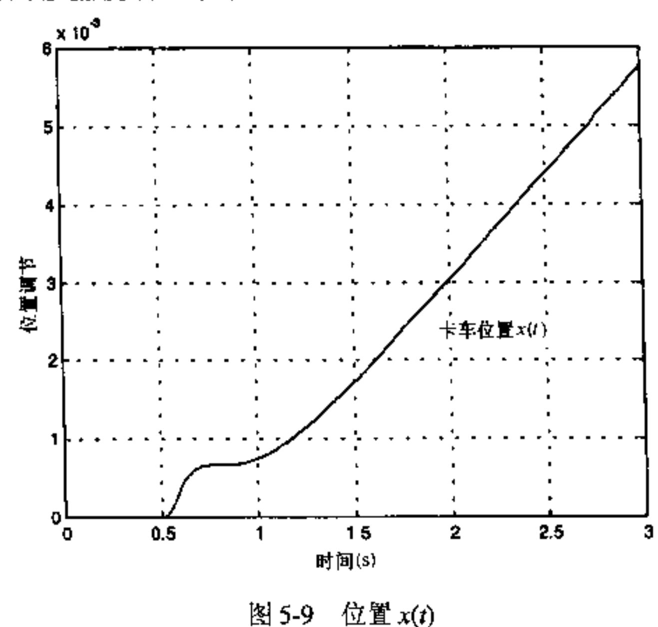


图 5-9 位置 $x(t)$

吊车的负位移是由于调节 $\theta(t)=0$ 的反应。

5.4 吊车位置 $x(t)$和角$\theta(t)$的调节

我们的目的是调节整个设备，因此需要一个位置微分反馈，在位置和脉冲干扰 $f(t)$处产生一个阶跃设定点（见图 5-10）。

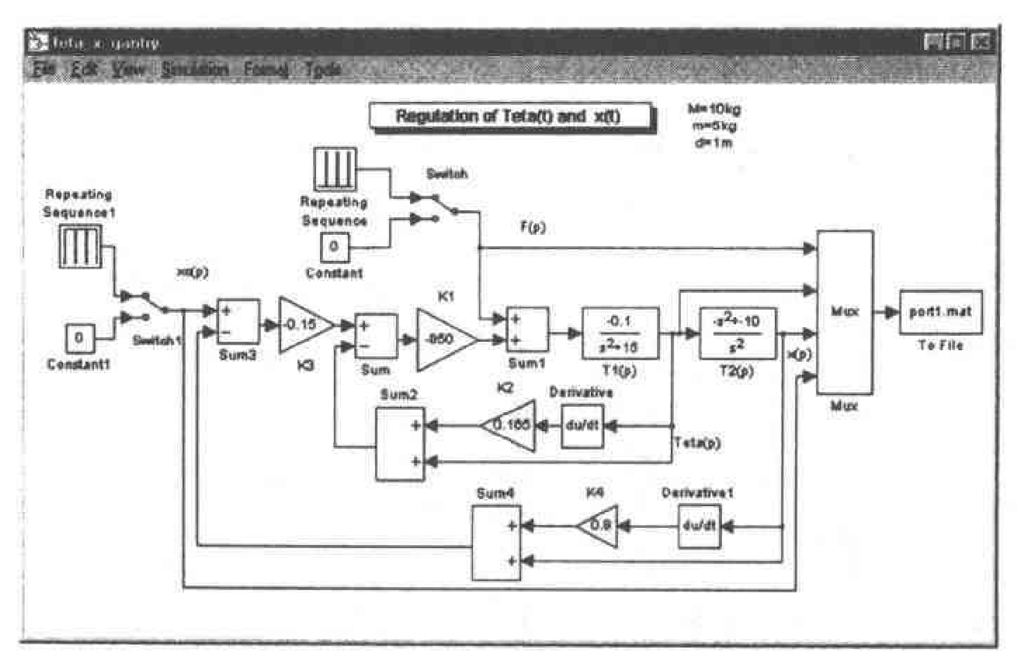


图 5-10 Teta(t)和 $x(t)$的调节图

为简化理论研究，并能选择参数 K_3 以及 K_4，须限制角度闭环传递函数表达式 $\dfrac{\theta(p)}{\theta_c(p)}$ 至它的静态增益：

$$\frac{\theta(p)}{\theta_c(p)}=\frac{K_1T_1(p)}{1+K_1(1+K_2p)T_1(p)}=\frac{K_1b_0}{a_0+K_1b_0}\times\frac{1}{1+\dfrac{K_1K_2b_0}{a_0+K_1b_0}p+\dfrac{1}{a_0+K_1b_0}p^2}$$

于是

$$\frac{\theta(p)}{\theta_c(p)}=\frac{K_1b_0}{a_0+K_1b_0}=K_\theta=0.85$$

◆ 闭环传递函数 $\dfrac{X(p)}{X_c(p)}$ 表达式

$$T_2(p)=\frac{X(p)}{\theta(p)}=-\frac{g+lp^2}{p^2}=-\frac{p^2+b_{01}}{p^2}$$

因为 l=1 m，限制在具有 b_0=−g=−10 的双积分表达式为

$$T_2(p)=\frac{b_{01}}{p^2}$$

因此

$$\frac{X(p)}{\theta(p)}=\frac{K_3K_\theta T_2(p)}{1+K_3K_\theta T_2(p)(1+K_4p)}=\frac{K_3K_\theta b_{01}}{p^2+K_3K_\theta K_4b_{01}p+K_3K_\theta b_{01}}$$

角频率 ω_n 和阻尼系数 ξ 为

$$\begin{cases}\omega_n=\sqrt{K_3K_\theta b_{01}}=\sqrt{-8.5K_3}\\ \xi=\dfrac{K_4\sqrt{K_3K_\theta b_{01}}}{2}=\dfrac{K_4\sqrt{-8.5K_3}}{2}\end{cases}$$

为确保运动无振荡，取

$$\begin{cases} \xi = 0.7 \\ \omega_n = 1\text{rad/s} \end{cases}$$

计算后得到 K_3 =−0.15、K_4 =1.4。然而，考虑到模型中的近似，这些结果必须为模拟调节的第一近似值。以如下结果作为结束：

$$K_3=-0.15 \qquad K_4=0.8$$

文件 gantry3.m 给出在阶跃设定点 $x(t)$和脉冲干扰$f(t)$下的 $x(t)$和 $\theta(t)$响应。

gantry3.m file

```
% Gantry in closed loop - Laplace's method
% Both Teta(t) and de x(t) régulation.
load port1.mat
t = signals(1,:); fp = signals(2,:); teta = signals(3,:);
x = signals(4,:); xc = signals(5,:);
figure(1), hf = line(t,fp);
xlabel('Time in seconds')
ylabel('Force in N'), axis([0 60 -2.5 1.5])
axet = axes('Position',get(gca,'Position'),...
          'XAxisLocation','bottom',...
          'YAxisLocation','right','Color','none',...
          'XColor','k','YColor','k');
ht = line(t,teta,'color','r','parent',axet);
ylabel('angle evolution'), grid
title('F(t) disturbance in N and \Theta(t) response in rd')
gtext('\leftarrow F(t) disturbance')
gtext('\leftarrow \theta(t) angle')
figure(2), plot(t,xc),hold on, grid, plot(t,x),hold off
title(' xc(t) set-point and x(t) position in m')
ylabel('regulation of the position')
xlabel('Time in seconds')
gtext('\leftarrow xc(t) set-point')
gtext('\leftarrow x(t) position')
```

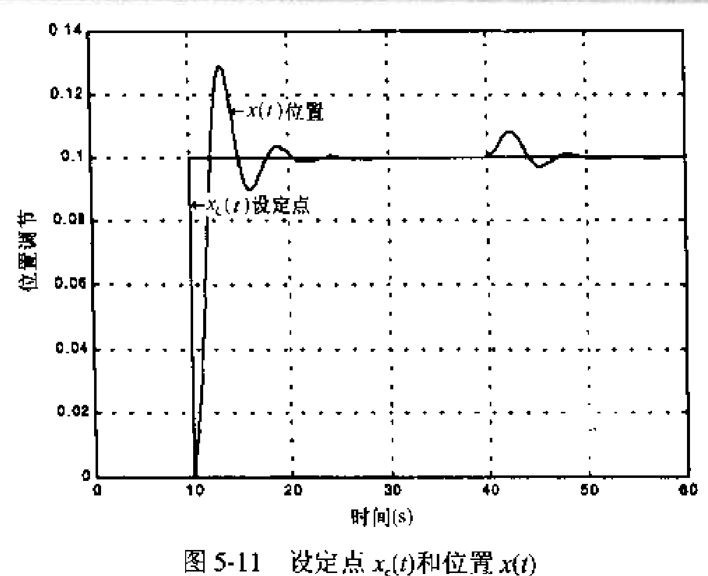


图 5-11　设定点 $x_c(t)$和位置 $x(t)$

我们看到，在开始时，吊车有一个与设定点方向相反的移动，这是 $\theta(t)$调节的结果。

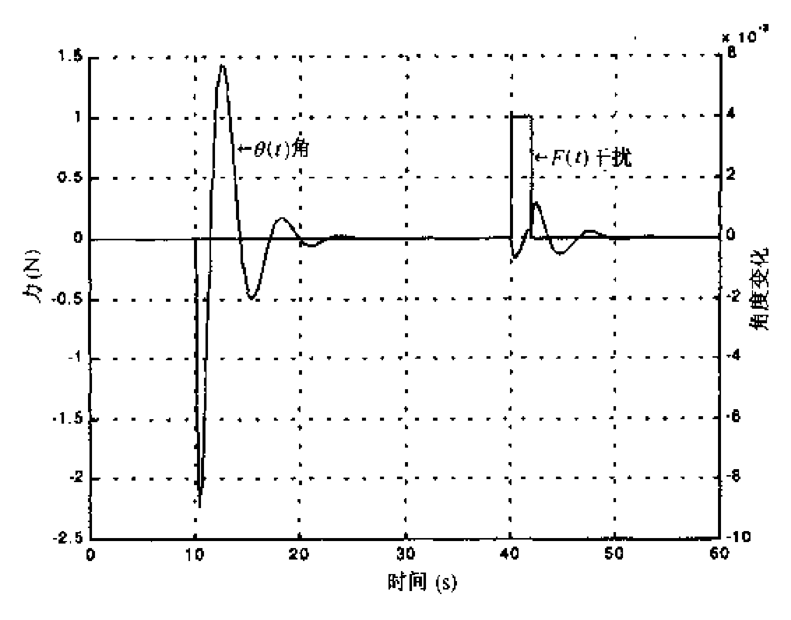


图 5-12　干扰 $F(t)$和响应$\theta(t)$

θ角的最大偏离在 x 阶跃位置为 0.1 m 时为−0.02 rad。1 N 的干扰 F 很快被抑制掉。

5.5　状态空间模型

移动高架吊车模型由以下微分方程组成：

$$\begin{cases} \ddot{\theta}(t) = -g\dfrac{(M+m)}{Ml}\theta(t) - \dfrac{1}{Ml}F(t) \\ \ddot{x}(t) = g\dfrac{m}{M}\theta(t) + \dfrac{1}{M}F(t) \end{cases}$$

通过选择状态矢量 $X = [x \quad \dot{x} \quad \theta \quad \dot{\theta}]^{\mathrm{T}}$，有如下状态方程：

$$\begin{bmatrix} x \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & g\dfrac{m}{M} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -g\dfrac{(M+m)}{Ml} & 0 \end{bmatrix} \times \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{1}{M} \\ 0 \\ -\dfrac{1}{Ml} \end{bmatrix} \times F(t)$$

观测方程为：

$$Y = \begin{bmatrix} x \\ \theta \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix}$$

◆　SIMULINK 模型（见图 5-13）

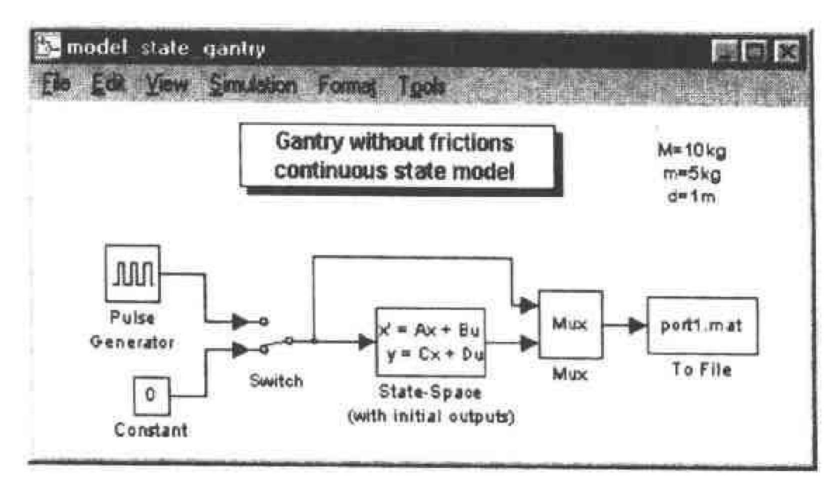


图 5-13　无摩擦连续状态吊车模型图

通过显示阶跃力 $F(t)$响应来检验该模型。

gantry11.m file

```
% Gantry in open loop
% analogical state model
% reading the simulation results data file
load port1.mat
t = signals(1,:); F = signals(2,:); x = signals(3,:);
teta = signals(4,:);
% drawing F(t) et Teta(t)curves
figure(1), hf = line(t,F);
xlabel('time in seconds'), ylabel('Force in N')
axis([0 7 -0.1 1.1])
axet = axes('Position',get(gca,'Position'),'XAxisLocation','bottom',...
    'YAxisLocation','right','Color','none',...
    'XColor','k','YColor','k');
ht = line(t,teta,'color','r','parent',axet);
ylabel('\theta(t) angle evolution'), grid
title('F(t) control in N and \Theta(t) response in rd')
gtext('F(t)\rightarrow'), gtext('\leftarrow \theta(t)')

% drawing F(t) et x(t) curves
figure(2)
plot(t,F),hold on, plot(t,x), hold off, grid
title('F(t) control and x(t) response')
ylabel('gantry in open loop')
xlabel('time in seconds')
gtext('Force in N'), gtext('x position in m')
```

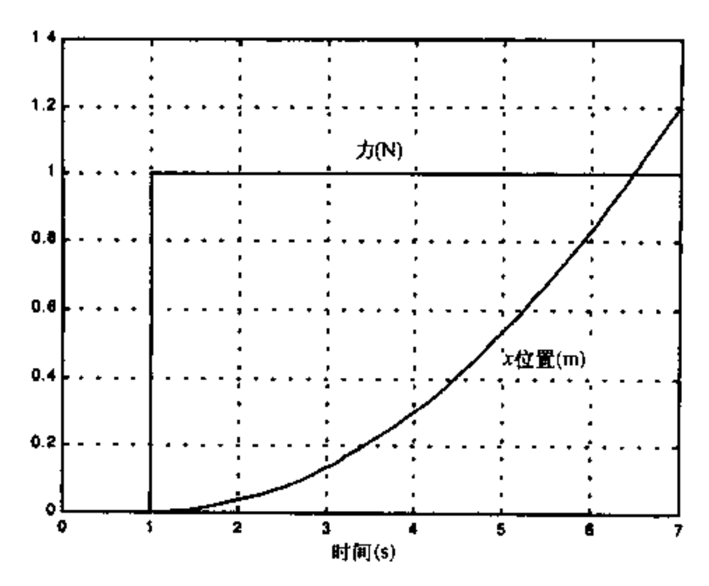


图 5-14　控制 $F(t)$和响应 $x(t)$

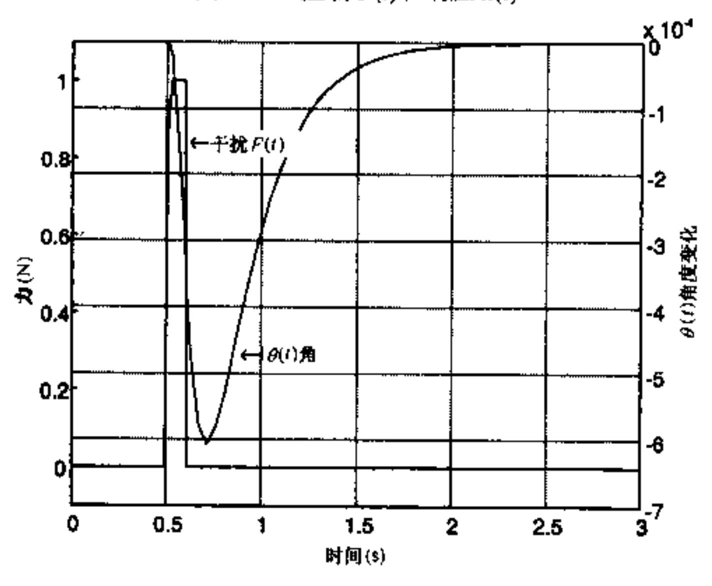


图 5-15　$F(t)$和$\theta(t)$图

得到的响应与传递函数模型一样。同样可以在选择的初始状态下看到模型的响应，在 SIMULINK 状态模型中设置它们，将手动开关放在常数 F=0 控制处。考虑对应于 x 为 0.1 m、θ 为-0.1 rad 下的初始状态矢量 $X_0 = \begin{bmatrix} 1 \\ -0.1 \end{bmatrix}$。由如下文件 gantry12.m 而得到初始状态响应。

gantry12.m file

```
% Portico in open loop
% analogical state model, response to initial conditions
% simulation results file reading
load port1.mat
t = signals(1,:); x = signals(3,:); teta = signals(4,:);
% Teta(t) drawing
figure(1), plot(t,teta), grid
xlabel('Time in seconds')
ylabel('angle evolution')
```

```
title('\Theta(t) response in rd')
gtext('\leftarrow \theta(t)')
% drawing of x(t)
figure(2), plot(t,x), grid, title('x(t) response')
ylabel('Portico in open loop')
xlabel('time in seconds')
gtext('x Position in m')
```

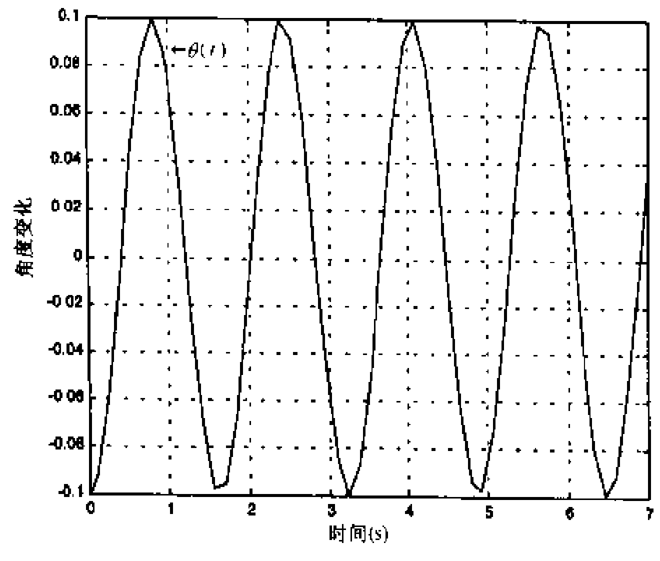


图 5-16 响应$\theta(t)$

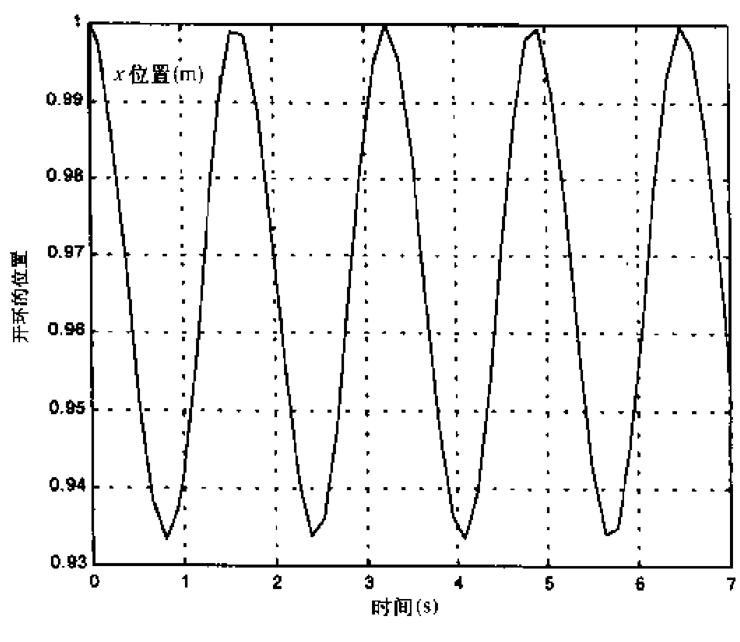


图 5-17 $x(t)$ 响应

可以看到，$\theta(t)$初始值取−0.1 rad，导致悬挂物振荡影响吊车的位置$x(t)$，它的初始值为 1 m。

5.5.1 离散状态空间模型

现在调节在离散状态空间下的移动高架吊车。过程的主要时间常数τ约为 1.6 s，固定采样周期为 0.1 s，即τ/16。

由于命令 c2dt，文件 gantry13.m 将连续状态空间模型转换为离散状态空间模型，接着根据阶跃力 $f(t)$画出 $x(t)$和 $\theta(t)$的响应。

gantry13.m file

```
% gantry crane without friction, in open loop
% Discrete state space modelling.

% Parameters initialisation
M = 10; m = 5;
l = 1; g = 10;
k1 = m*g/M;
k2 = -(M+m)*g/(M*l);
k3 = 1/M;
k4 = -1/(M*l);
% State space matrices initialisation
A = [0 1 0 0 ;0 0 k1 0 ; 0 0 0 1 ; 0 0 k2 0 ];
B = [0; k3; 0; k4];

% Initialisation of observation equation matrices
C = [1 0 0 0; 0 0 1 0];
D = 0;

% Conversion of the continuous state space model to the
% discrete one at fe = 10Hz
Te = 0.1 ;
[Ad,Bd,Cd,Dd] = c2dt(A,B,C,Te,0);

% State initial conditions
x(:,1) = [0 0 0 0]';

% Control signal
N = 100;

i = 1:N
f = zeros(size(i));
f = (i>11) ;

% System's response

for i = 2:N
  x(:,i) = Ad*x(:,i-1)+Bd*f(i-1);
end;

% x(t) and teta(t) plotting

figure(1)
hf = line(1:N,f(:));
xlabel('Samples')
ylabel('Force in N')
axis([0 100 -0.1 1.1])
axet = axes('Position',get(gca,'Position'),...
    'XAxisLocation','bottom',...
    'YAxisLocation','right','Color','none',...
```

```
    'XColor','k','YColor','k');
ht=line(1:N,x(1,:),'color','r','parent',axet);
ylabel('Evolution of the position');
title('x(t) response in m to a step force f(t)=1 N');
gtext('x(t)')
gtext('f(t)')

figure(2)
hf = line(1:N,f(:));
xlabel('Samples')
ylabel('Force in N')
axis([0 100 -0.1 1.1])

axet = axes('Position',get(gca,'Position'),'XAxisLocation','bottom',...
     'YAxisLocation','right','Color','none',...
     'XColor','k','YColor','k');
ht = line(1:N,x(3,:),'color','r','parent',axet);
ylabel('\theta(t) angle evolution'), grid
title('\Theta(t) response in rd to a step f(t)=1 N')
gtext('f(t)\rightarrow')
gtext('\leftarrow \theta(t)')
```

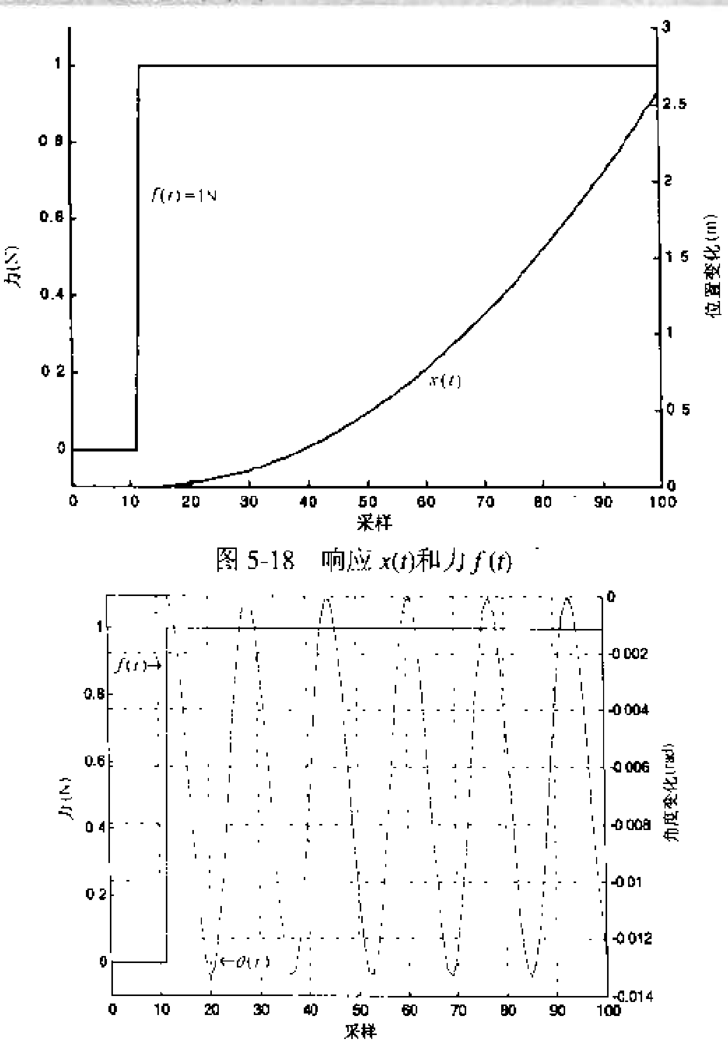


图 5-18　响应 $x(t)$和力 $f(t)$

图 5-19　响应$\theta(t)$和力 $f(t)$

5.5.2 Luenberger 状态观测器

假定过程是确定性的，目标是从完整的状态估计中得到状态矢量。

首先用 obs 观测矩阵来判断过程的可观测性。矩阵可写为

$$\text{obs} = \begin{bmatrix} Cd & CdAd & CdAd^2 & \cdots & CdAd^{n-1} \end{bmatrix}$$

它可被这样计算：

```
% Observability matrix
n = 4;    % order of Ad matrix
l = 2;    % Cd Matrix lines number

for i = 0:n-1,
   x = Cd*(Ad^i);
   for j = 1:l,
     obs (2*i+j,:) = x(j,:);
   end
end

% plotting the observability matrix
printmat (obs,'Observability matrix')
Observability matrix =
           ----1----          ----2----     ----3----      ----4----
--1-->     1.00000               0              0              0
--2-->        0                  0          1.00000            0
--3-->     1.00000           0.10000        0.02469        0.00083
--4-->        0                  0          0.92593        0.09752
--5-->     1.00000           0.20000        0.09510        0.00647
--6-->        0                  0          0.71470        0.18059
--7-->     1.00000           0.30000        0.20080        0.02103
--8-->        0                  0          0.39760        0.23691
```

接着通过计算观测矩阵的秩来判断过程是否可观测。在本例中，秩为 4，并使用 rank 命令。

```
% rank calculation
disp (['rank = 'num2str(rank(obs))])

rank=4
```

如果有“控制系统工具箱”，命令 obsv(Ad,Cd)可从过程矩阵 Ad 和 Cd 中计算矩阵的可观测性。这里，用文件 gantry14.m 可证明过程的可观测性。

gantry14.m file

```
% Travelling gantry crane in open loop
% Observability verification

% State model initialisation
M=10; m=5; l=1; g=10;
k1=m*g/M;
k2=-(M+m)*g/(M*l);
k3=1/M;
k4=-1/(M*l);
```

```
A=[0 1 0 0 ;0 0 k1 0 ; 0 0 0 1 ; 0 0 k2 0 ];
B=[0; k3; 0; k4];
C=[1 0 0 0; 0 0 1 0];

% Discrete state model
Te=0.1;
[Ad,Bd,Cd,Dd]=c2dt(A,B,C,Te,0);

% Observability matrix calculation
obs=obsv(Ad,Cd);

% Plotting of observability matrix
printmat(obs,'observability matrix');

% rank calculation
disp(['rank= 'num2str(rank(obs))]);
if rank(obs)==4 disp('the process is observable')
else disp('the process is not observable')
end;
```

```
>> gantry14

observability matrix =
                --1--          --2--         ----3----        ----4----
--1-->      1.00000              0                0                0
--2-->            0              0          1.00000                0
--3-->      1.00000        0.10000          0.02469          0.00083
--4-->            0              0          0.92593          0.09752
--5-->      1.00000        0.20000          0.09510          0.00647
--6-->            0              0          0.71470          0.18059
--7-->      1.00000        0.30000          0.20080          0.02103
--8-->            0              0          0.39760          0.23691
```

rank=4，过程是可观测的。

◆ 状态估计的 SIMULINK 模型（见图 5-20）

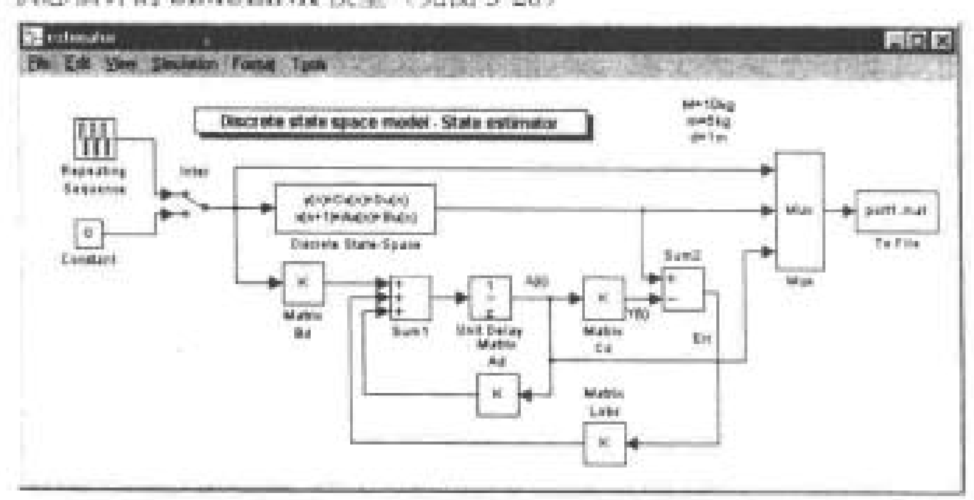


图 5-20 离散状态空间模型-状态估计器

固定状态估计器动态的增益矩阵名为 Lobs。为确保根据系统时间响应的状态矢量能迅

速收敛，必须选择该矩阵。但是，必须确保估计器的稳定性，这意味着$[Ad \quad - Lobs\, Cd]$矩阵的方程特征根必须在单位圆内。观测器必须对修正系统做出快速响应，这使特征值小于修正系统的值。修正系统的动态是$[Ad - K\, Bd]$矩阵的特征方程的函数，K 代表调节系统的状态反馈矩阵。于是必须选择修正系统的极点a_i：

$$\phi = \det[Ad - K.Bd] = z^n + a_{n-1}.z^{n-1} + \cdots + a_0$$

要得到具有特征方程$(z-0.6)^4$的修正过程，于是四重极点等于 0.6，这对应于如下的修正系统特征方程：

$$\phi_c = z^4 - 2.4z^3 + 2.16z^2 - 0.864z + 0.1296$$

对于状态估计器，要得到合适的角频率ω_n=2π rad/s 和阻尼系数ξ=0.707 的二阶动态系统：

$$\phi_r = z^2 + p_1 z + p_2$$

且

$$p_1 = -2e^{-\xi\omega_n T_e}\cos(\omega_n T_e\sqrt{1-\xi^2})$$
$$p_2 = e^{-2\xi\omega_n T_e}$$

因此

$$\phi_r = z^2 - 1.158z + 0.4112$$

为方便计算，将状态方程写成正则或伴随型：

已知

$$Ad = \begin{bmatrix} 1 & 0.1 & 0.0247 & 0.0008 \\ 0 & 1 & 0.4876 & 0.0247 \\ 0 & 0 & 0.9259 & 0.0975 \\ 0 & 0 & -1.4628 & 0.9259 \end{bmatrix} \quad 和 \quad Bd = \begin{bmatrix} 0.0005 \\ 0.0099 \\ -0.0005 \\ -0.0098 \end{bmatrix}$$

因此过程特征方程写为：

$$\det[zI - Ad] = \phi = z^4 - 3.85z^3 + 5.7z^2 - 3.84z + 1$$

可用来计算允许伴随型的传递矩阵Wc。

文件 gantry15.m 可用来计算：

gantry15.m

```
% State return control
M = 10; m = 5;
d = 1;
g = 10;
k1 = m*g/M;
k2 = -(M+m)*g/(M*d);
k3 = 1/M;
k4 = -1/(M*d);
Te = 0.1;
Dd = 0;
A = [0 1 0 0 ;0 0 k1 0 ; 0 0 0 1 ; 0 0 k2 0 ];
B = [0; k3; 0; k4];
C = [1 0 0 0; 0 0 1 0];
[Ad,Bd,Cd,Dd] = c2dt(A,B,C,Te,0);
com = ctrb(Ad,Bd);
% characteristics equation of the state estimator
```

```
m = sqrt(2)/2;
wn = 2*pi;
p1 = -2*exp(-m*wn*Te)*cos(wn*Te*sqrt(1-m^2));
p2 = exp(-2*m*wn*Te);
p = [0 0 1 p1 p2];
% characteristics equation of the process
phi = poly(Ad);
% companion form of the state equation
% Wc transformation matrix
n = 4;
Wc(:,4) = Bd;
for i = 1:n-1,
  Wc(:,n-i) = Ad*Wc(:,n-i+1)+phi(n-i+1)*Bd;
end

% Plotting the transformation matrix
printmat(Wc,'transformation matrix')
```

```
transformation matrix =
           ----1----    ----2----     ----3----     ----4----
--1-->     0.00050     -0.00040      -0.00040      0.00050
--2-->    -0.01001      0.02824      -0.02827      0.01008
--3-->     0.00049     -0.00050      -0.00049      0.00049
--4-->    -0.00968      0.02922      -0.02925      0.00975
```

```
% companoion form of the matrices
Ac = inv(Wc)*Ad*Wc;
Bc = inv(Wc)*Bd;
% Plotting the companion form matrices
printmat(Ac,'companion matrix Ac')
printmat(Bc,'companion matrix Bc')
companion matrix Ac =
           ----1----    ----2----        ----3----        ----4----
--1-->     0.00055      1.00000          5.32907e-015     -7.54952e-015
--2-->    -0.00213     -5.32907e-015     0.00000          -7.10543e-015
--3-->    -0.00075     -3.55271e-015     3.55271e-015      1.00000
--4-->    -0.98864      3.84130         -5.6995            3.85132
companion matrix Bc =
              ----1----
      --1-->  -1.77636e-015
      --2-->  -1.77636e-015
      --3-->  0
      --4-->  1.00000
```

可以看到：

$$Lobs' = \begin{bmatrix} l_0 & l_0 \\ l_1 & l_1 \\ l_2 & l_2 \\ l_3 & l_3 \end{bmatrix}$$

给出：

$$\begin{cases} l_3 = 0 + 3.85 = 3.85 \\ l_2 = 1 - 5.7 = -4.7 \\ l_1 = -1.158 + 3.84 = 2.682 \\ l_0 = 0.411 - 1 = -0.589 \end{cases}$$

已知 $Lobs_c = LobsW_c^{-1}$，得到估计器的增益矩阵。

```
% Observability matrix
n = 4;   % order of Ad matrix
l = 2;   % Cd Matrix lines number
for I = 0:n-1
   x = Cd*(Ad^i);
   for j = 1:l
      obs(2*i+j,:) = x(j,:);
   end
end
% plotting the observability matrix
printmat(obs, 'observability matrix')
companion matrix Lobsc =
               ----1----    ----2----
      --1-->  12689.77927  12689.77927
      --2-->   2999.46829   2999.46829
      --3-->   9492.12236   9492.12236
      --4-->   2822.66759   2822.66759
```

文件 gantry16.m 可检查状态估计的正确性。

gantry16.m

```
% Travelling gantry crane in open loop
% state estimator
load port1.mat
t = signals(1,:);
f = signals(2,:);
x = signals(3,:);              % x(t) signal
teta = signals(4,:);           % teta(t) signal
x_rec = signals(5,:);          % x(t) reconstructed signal
dx_rec = signals(6,:);         % x'(t) reconstructed signal
teta_rec = signals(7,:);       % teta(t) reconstructed signal
dteta_rec = signals(8,:);      % teta'(t) reconstructed signal

% f(t) control and x(t) position plotting
figure(1)
hf = line(t,f);
xlabel('time in seconds')
ylabel('Force in N')
axis([0 20 -1.1 1.1])
axet = axes('Position',get(gca,'Position'),'XAxisLocation','bottom',...
    'YAxisLocation','right','Color','none',...
    'XColor','k','YColor','k');
ht = line(t,x,'color','r','parent',axet);
ylabel('x(t) position evolution')
grid
title('F(t) command in N and x(t) response in m')
gtext('f(t)\rightarrow')
```

```
gtext('\leftarrow x(t)')

% Plotting of reconstructed x(t) and x'(t)
figure(2)
hf = line(t,x_rec);
xlabel('time in seconds')
ylabel(' xr(t) Position ')

axet = axes('Position',get(gca,'Position'),...
   'XAxisLocation','bottom',...
   'YAxisLocation','right','Color','none',...
   'XColor','k','YColor','k');
ht = line(t,dx_rec,'color','r','parent',axet);
ylabel(' xr''(t) derivative of the position ')
grid
title('reconstructed position signals')
gtext('xr(t)\rightarrow'), gtext('\leftarrow xr''(t)')
% Plotting of reconstructed teta(t) and teta'(t)
figure(3)
hf = line(t,teta_rec);
xlabel('time in seconds')
ylabel('\thetar(t) angle')
axet = axes('Position',get(gca,'Position'),...
          'XAxisLocation','bottom',...
          'YAxisLocation','right','Color','none',...
          'XColor','k','YColor','k');
ht = line(t,dteta_rec,'color','r','parent',axet);
ylabel(' \thetar''(t) angle derivative')
grid
title('angular reconstructed signals')
gtext('\thetar(t)\rightarrow')
gtext('\leftarrow\thetar''(t)')
```

◆ 控制 $f(t)$ 和位置 $x(t)$

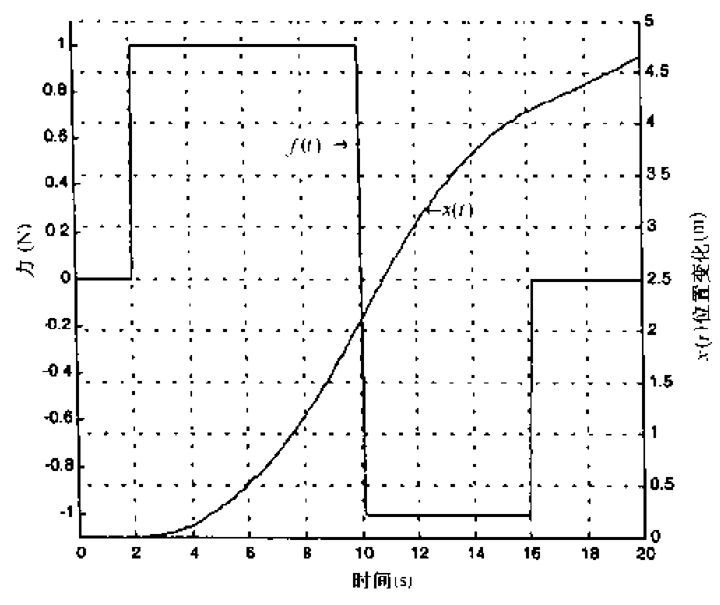


图 5-21　控制 $f(t)$ 和响应 $x(t)$

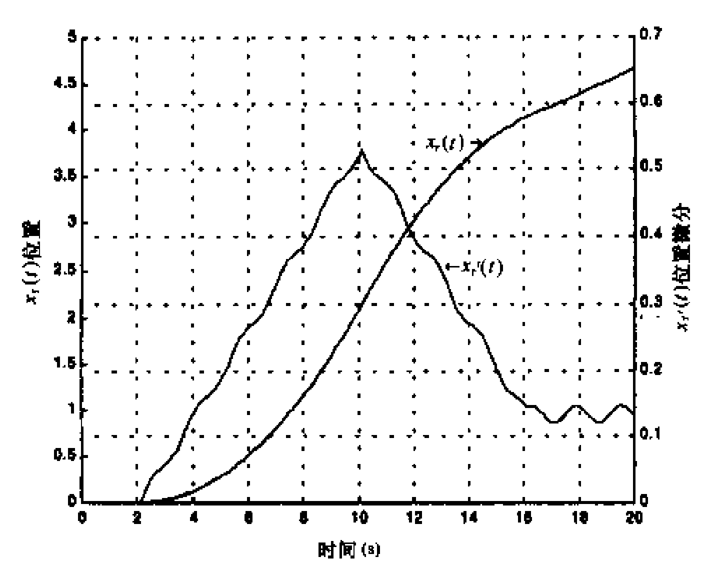


图 5-22　位置信号的重构

位置的微分清楚地显示了表达式 $x(t)$中的正弦函数。

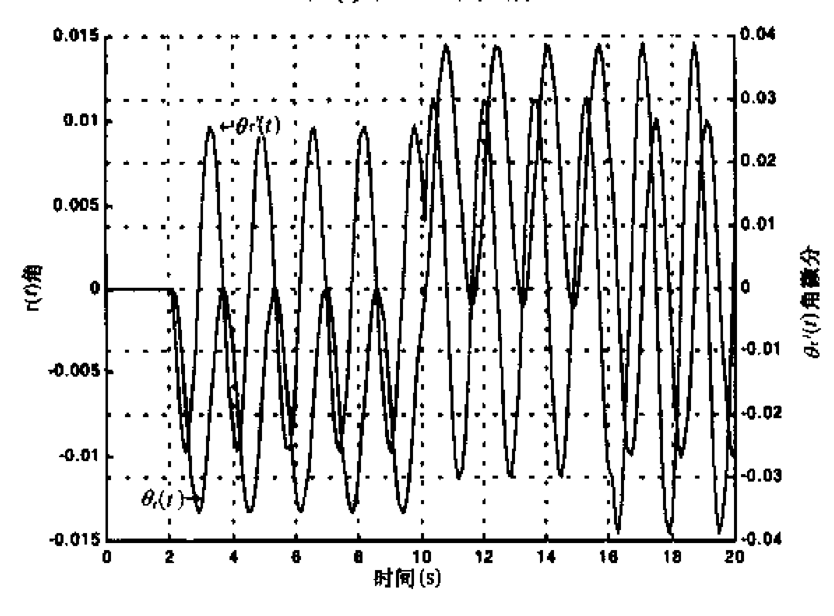


图 5-23　角度信号重构

由此很清楚地得到 $x(t)$和$\theta(t)$信号以及它们的微分。

5.5.3　过程的状态空间控制

首先检查过程是否良好。为此，需计算可控性矩阵 *com*。可控性矩阵记为

$$com = \left[Bd \quad AdBd \quad Ad^2Bd \quad \cdots \quad Ad^{n-1}Bd \right]$$

它可进行如下计算。

```
% controllability matrix calculation
n = 4

for i = 0:n-1,
    com(:,i+1) = (Ad^i)*Bd;
```

```
end

% plitting of controllability matrix
printmat(com,'Controllability matrix')

Controllability matrix =
              ----1----    ----2----    ----3----    ----4----
--1-->         0.00050      0.00147      0.00237      0.00317
--2-->         0.00992      0.00944      0.00854      0.00737
--3-->        -0.00049     -0.00141     -0.00211     -0.00251
--4-->        -0.00975     -0.00831     -0.00563     -0.00212
```

然后通过计算可控性矩阵的秩来检查过程是否良好。在此过程中，矩阵的秩为 4。

```
% rank calculation
disp (['rank = 'num2str(rank(com))])

rank = 4
```

过程是良好的。

如果有“控制系统工具箱”，ctrb(*Ad*,*Bd*)从过程的 *Ad* 和 *Bd* 矩阵来计算可控制矩阵。

```
Com = ctrb(Ad,Bd);
Printmat (com,'Controllability matrix')

Controllability matrix =
              ----1----    ----2----    ----3----    ----4----
--1-->         0.00050      0.00147      0.00237      0.00317
--2-->         0.00992      0.00944      0.00854      0.00737
--3-->        -0.00049     -0.00141     -0.00211     -0.00251
--4-->        -0.00975     -0.00831     -0.00563     -0.00212
```

◆ SIMULINK 模型（见图 5-24）

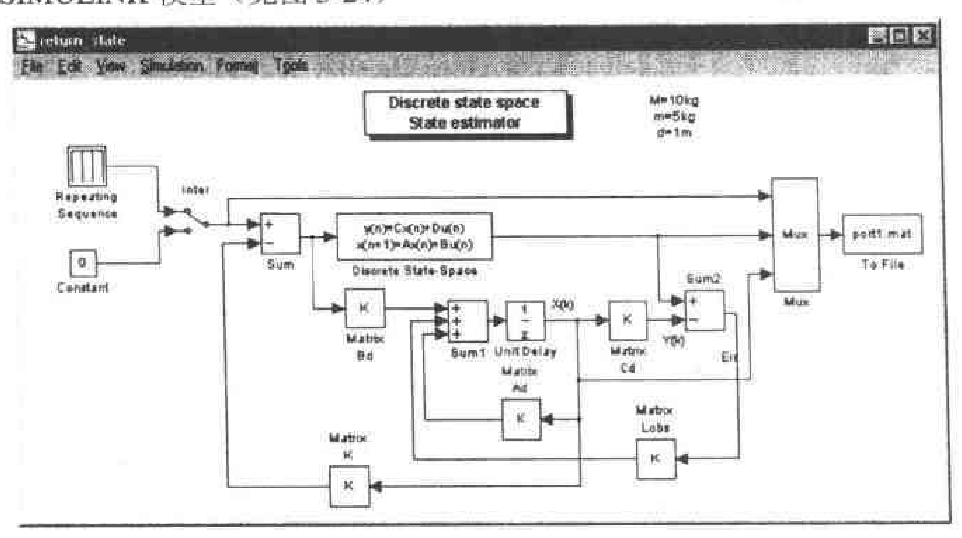


图 5-24 离散状态空间状态估计器

在仿真此模型前，必须计算状态反馈系数矩阵 $\boldsymbol{K}$:

$$\boldsymbol{K}=\begin{bmatrix}k_0 & k_1 & k_2 & k_3\end{bmatrix}$$

因为要得到如下控制系统的特征方程式

$$\phi_c = z^4 - 2.4z^3 + 2.16z^2 - 0.864z + 0.1296$$

所以得到：

$$\begin{cases} k_3 = -2.4 + 3.8513 = 1.4513 \\ k_2 = 2.16 - 5.7 = -3.54 \\ k_1 = -0.864 + 3.8413 = 2.9773 \\ k_0 = 0.1296 - 0.98864 = -0.759 \end{cases}$$

已知 $K_c = KW_c^{-1}$，得到如下反馈矩阵系数。

gantry17.m

```
% state return matrix
n = 4;          % vector order
phi = poly(Ad);
phic = [1 -2.4 2.16 -0.864 0.1296];
for i = 1:n
  K(1,n-i+1) = phic(1,i+1)-phi(1,i+1);
end
Kc = K*inv(Wc);
printmat(Kc,'state feedback coefficients Kc matrix')

state feedback coefficients Kc matrix =
           ----1----    ----2----    ----3----    ----4----
--1-->    259.22414    220.34052   -327.38817    105.01043
```

文件 gantry18.m 可得到控制系统的阶跃响应。

gantry18.m

```
% Gantry in open loop
% return state control
load port1.mat
t = signals(1,:);
f = signals(2,:);               % F(t) control signal
x = signals(3,:);               % x(t) signal
teta = signals(4,:);            % teta(t) signal
fr = signals(5,:);              % Fr(t) state feedback control

% F(t) control and
figure(1)
hf = line(t,f);
xlabel('time in seconds')
ylabel('step force in N')
axis([0 20 -0.1 1.1])
axet = axes('Position',get(gca,'Position'),'XAxisLocation','bottom',...
     'YAxisLocation','right','Color','none',...
     'XColor','k','YColor','k');
ht = line(t,fr,'color','r','parent',axet);
sys = zpk([],[0.75 0.75 0.75 0.75],1,Te);
[y,t] = lsim(sys,f,t);
coef = max(fr)/max(y);
hc = line(t,coef*y,'color','g','parent',axet);
ylabel('Corrected system response in N')
grid
title('step force response')
```

```
gtext('f(t)\rightarrow'), gtext('\leftarrow fr(t)')
gtext('\leftarrow suitable dynamics')

% x(t) plotting
figure(2), plot(t,x), grid
title(' x(t) truck position')
xlabel('Time'), ylabel('Position in m')
gtext('x(t)')

% teta(t) plotting
figure(3)
plot(t,teta), grid
title('\theta(t) angle'), xlabel('time')
ylabel('Angle in rd'), gtext('\theta(t)')
```

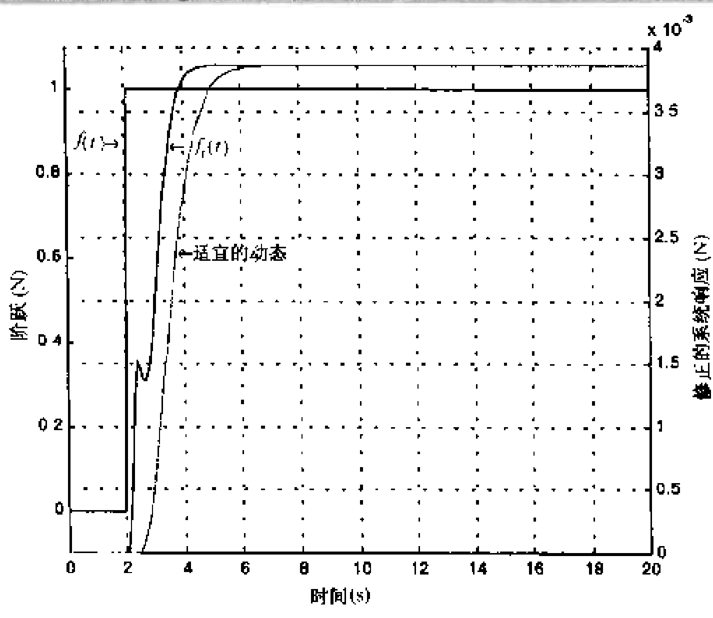


图 5-25　阶跃响应

系统的阶跃响应时间大约为 2 s。可以看出角 $\theta(t)$ 的变化影响了位置 $x(t)$。

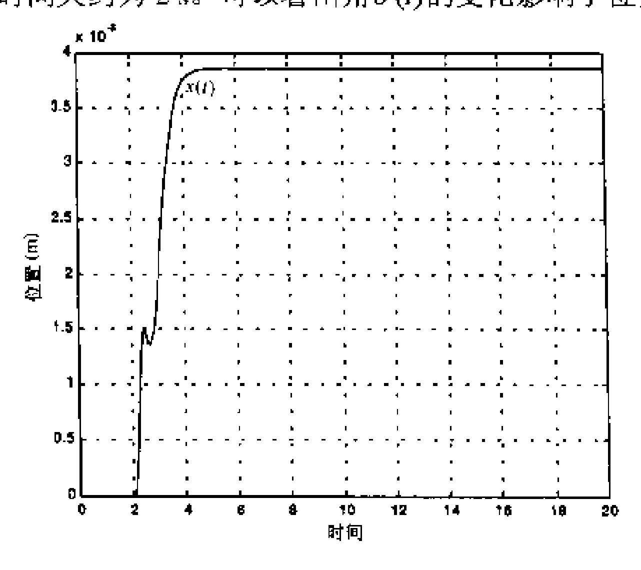


图 5-26　吊车位置 $x(t)$

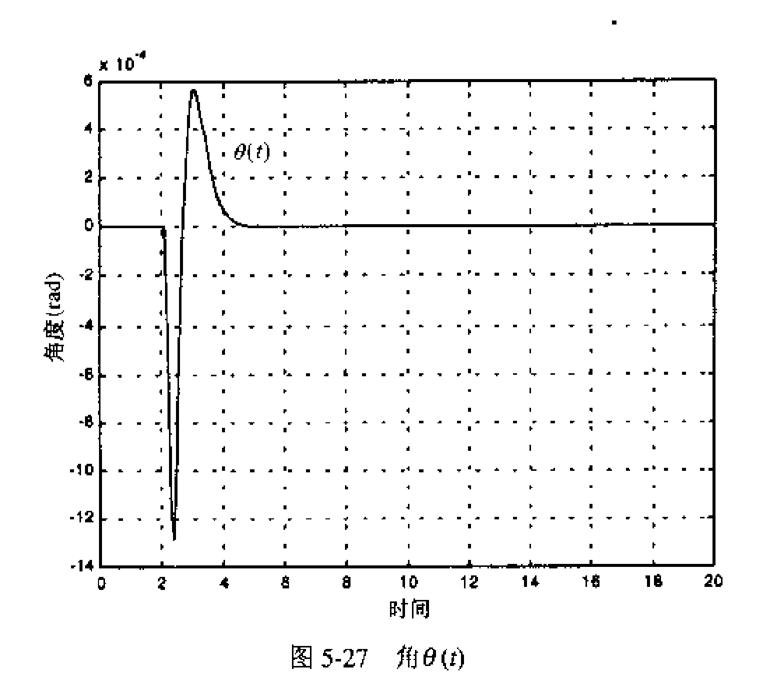


图 5-27　角 $\theta(t)$

悬挂物振荡的最大角偏离为 0.001 2 rad，这使微角度线性化近似有效。

5.5.4　加入积分修正

我们已经对控制系统的动态特性比较满意，现在需要控制 x 位置而无稳态误差。为此，在 x 控制环中加入一个积分修正。

◆　控制系统框图（见图 5-28）

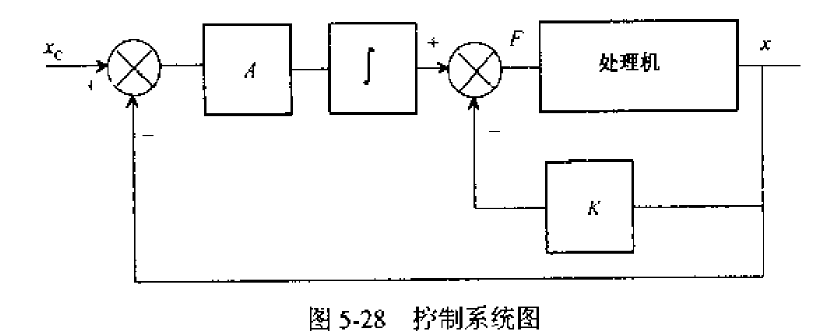


图 5-28　控制系统图

◆　对应的 SIMULINK 模型（见图 5-29）

为控制位置 $x(t)$，需摘录状态矢量的第 1 项以使之在控制规则中，为此用 demux 块和 ter 终端。终端 ter1、2、3 只产生目标，避免了由于非连贯 $\dot{x}$ 、θ 、$\dot{\theta}$ 输出的模拟中的 Warning。增益 A 固定在 100，确保 $x(t)$响应不超调。

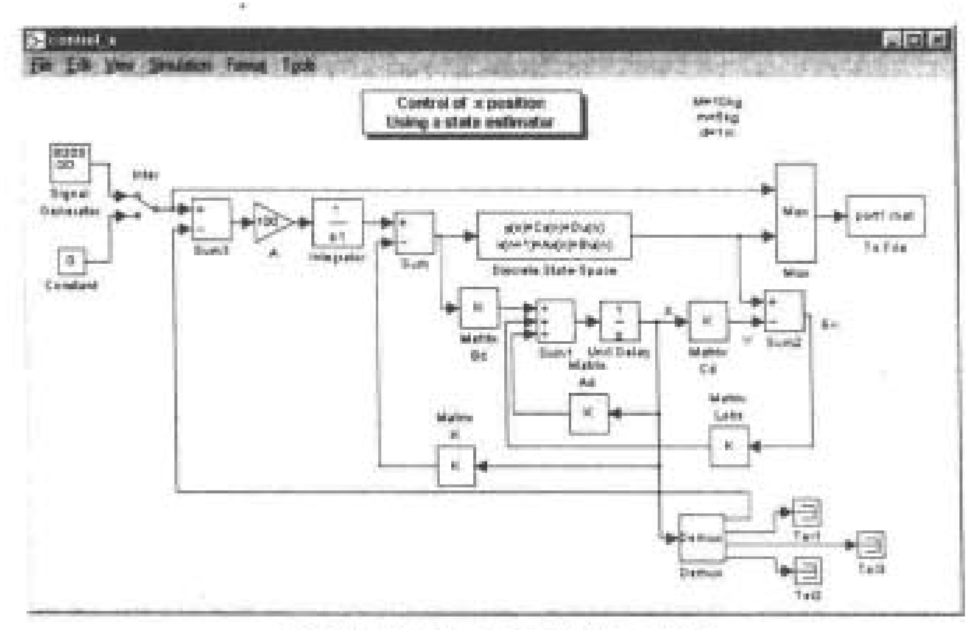


图 5-29　使用状态估计器的位置 x 控制

文件 gantry19.m 给出具有对最大值 3.5 m 的周期设定点位置的响应。

gantry19.m file

```
% Portico in closed loop
% control of x
load port1.mat
t = signals(1,:);
xc = signals(2,:);      % xc(t) control signal
x = signals(3,:);       % x(t) signal
teta = signals(4,:);    % teta(t) signal
% xc(t) and x(t) plotting
figure(1)
plot(t,x), hold on
plot(t,xc), hold off, grid
title(' xc(t) set-point position and x(t) response')
xlabel('Time')
ylabel('Position in m')
gtext('x(t)'),gtext('xc(t)')
% teta(t) plotting
figure(2)
plot(t,teta), grid
title('\theta(t) angle ')
xlabel('Time')
ylabel('Angle in rd')
gtext('\theta(t)')
```

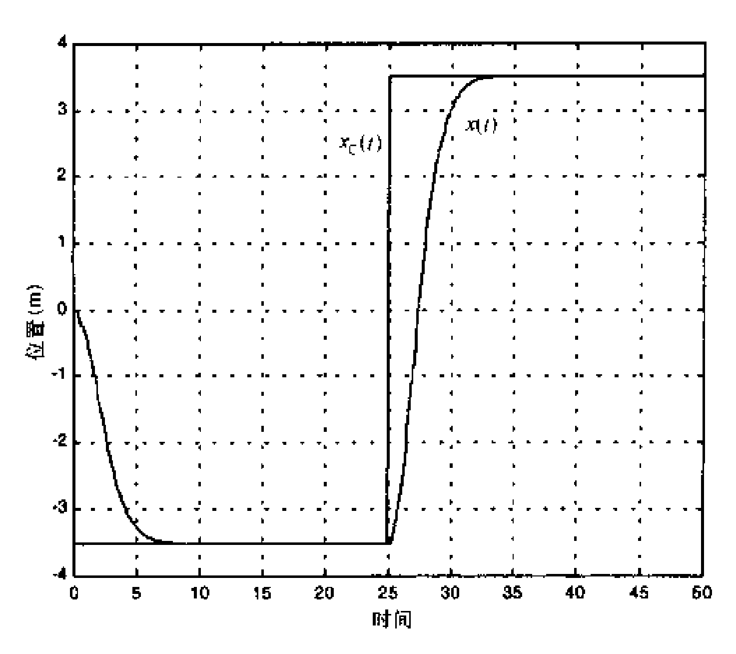


图 5-30　设定点位置 $x_c(t)$和响应 $x(t)$

位置达到时有一点超调，角$\theta(t)$变化小于 17°，这证明了小角度的近似（5 %精确度）。

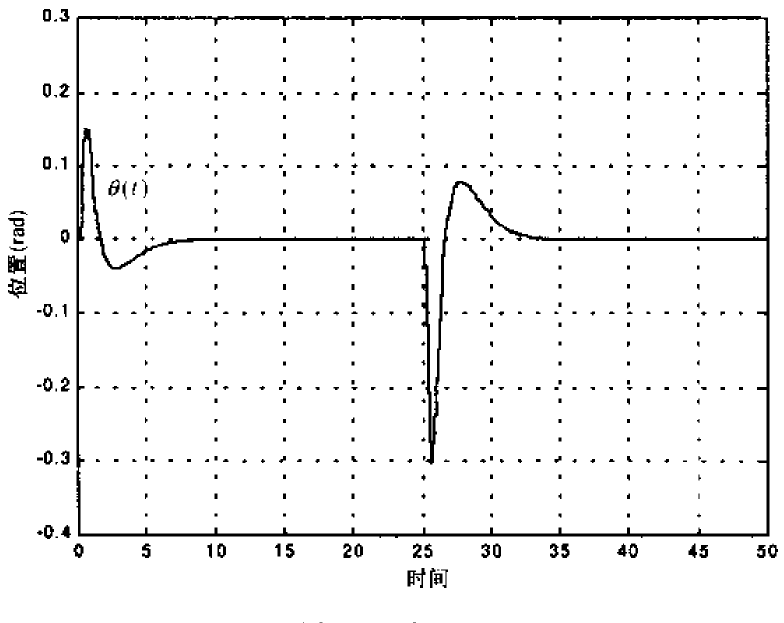


图 5-31　角$\theta(t)$

通过加一个干扰 $F_p(t)$，不仅在追踪模式而且在调整状态（干扰抑制）同样可检验控制系统的响应。

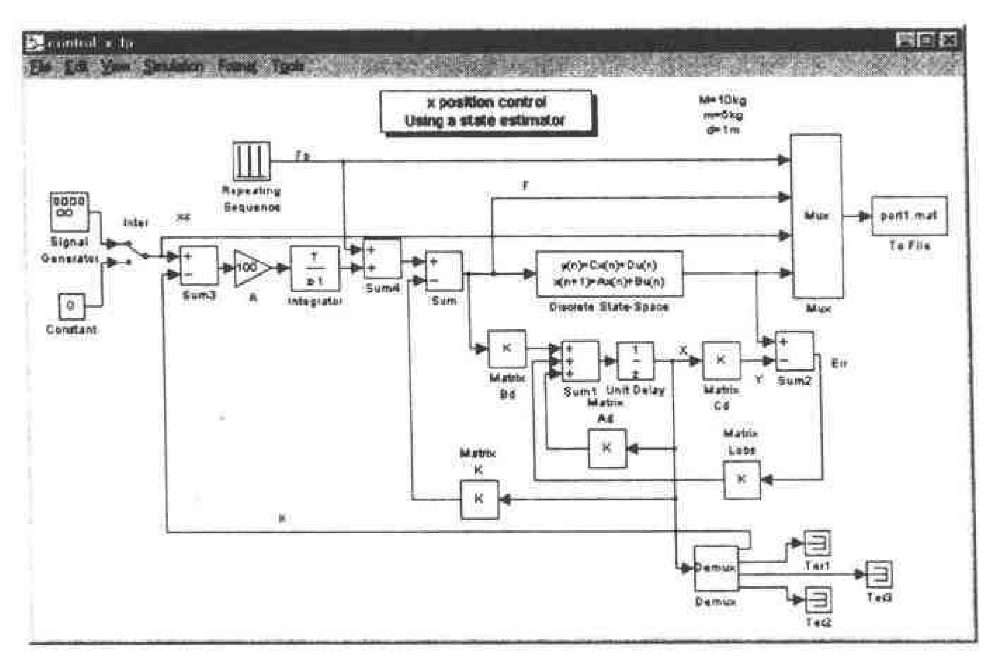


图 5-32 使用状态估计器的位置 x 控制

分别在 t=10 s 和 t=20 s 的离散时间加一个幅度为+10 N、持续时间为 1 s 的脉冲干扰。文件 gantry20.m 绘出了得到的结果。

gantry20.m

```
% Gantry in closed loop
% Control of x in regulation and pursuit
load port1.mat, t = signals(1,:);
fp = signals(2,:);        % Fp(t) disturbance signal
f = signals(3,:);         % F(t)control signal
xc = signals(4,:);        % xc(t) set-point signal
x = signals(5,:);         % x(t) position signal
teta = signals(6,:);      % teta(t) signal
% xc(t) and x(t) drawing
figure(1), plot(t,x), hold on, plot(t,xc), hold off, grid
title(' xc(t) set-point and x(t) response')
gtext('x(t)'), gtext('xc(t)')
xlabel('Time'), ylabel('Position in m')
% Fp(t) disturbance and F(t) control plotting
figure(2), hfp = line(t,fp);
xlabel('Time in seconds'),
ylabel('force disturbance in N'), axis([0 35 -1 11])
axet = axes('Position',get(gca,'Position'),...
          'XAxisLocation','bottom',...
          'YAxisLocation','right','Color','none',...
          'XColor','k','YColor','k');
hf = line(t,f,'color','r','parent',axet);
ylabel('f(t) process control in N'), grid
title('disturbance and force command')
gtext('fp(t)\rightarrow'), gtext('\leftarrow f(t)')
% teta(t) plotting
figure(3), plot(t,teta), grid
title('Angle of the suspended mass \theta(t)')
xlabel('Time'), ylabel('Angle in rd'), gtext('\theta(t)')
```

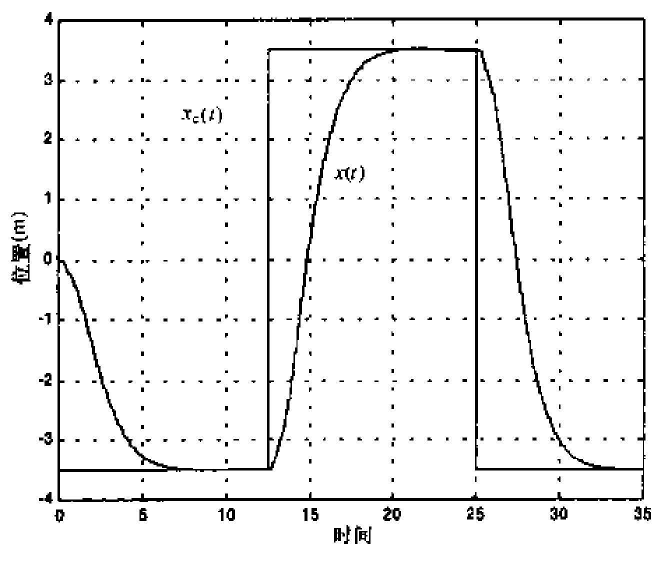


图 5-33　设定点 $x_c(t)$和响应 $x(t)$

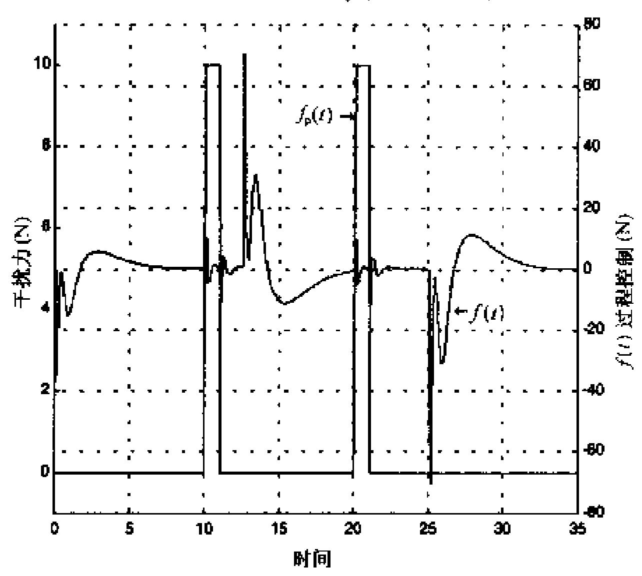


图 5-34　干扰和控制力

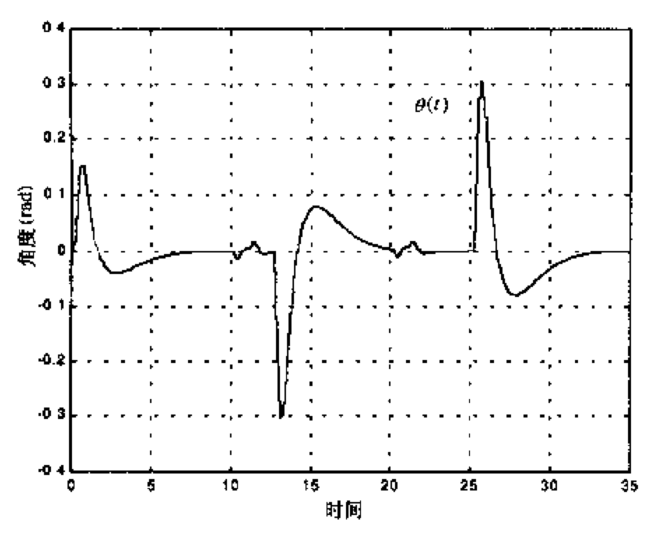


图 5-35　悬挂物的角度$\theta(t)$

干扰并不影响吊车位置，相反，它很小的作用却影响悬挂物的角度。

5.6 移动高架吊车的图形制作

在 SIMULINK 模型中加入调用名为 animat.m 的图形制作系统的 S 函数，可参看附录中的 S 函数。

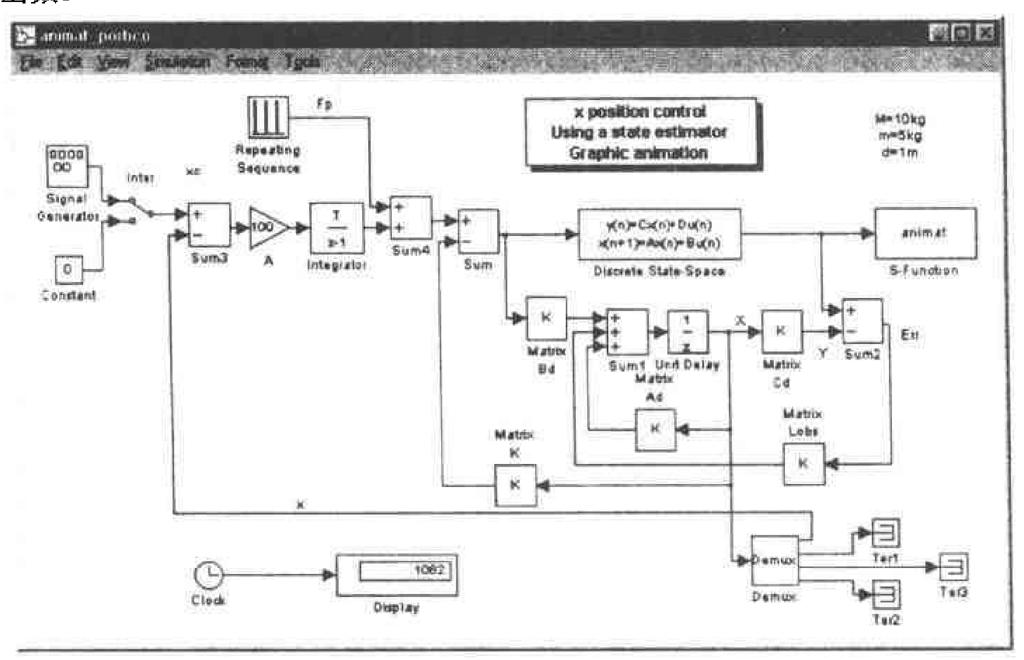


图 5-36 使用状态估计器图形制作的 x 位置控制

在函数输入时 x 和θ 作为参数，因此，为代表起重架，最好使用 plot 函数；“移动装置车+缆绳+悬挂物”由矢量对 Mx 和 My 表示。

在每个采样周期，这个矢量对重新实现一次（flag=2）。图中不移动部分在初始状态呈现（flag=0）。移动组成（矢量 Mx 和 My）在下面的图解中给出，这里采用 x 和 y 为 ± 50 m 的直角坐标系（见图 5-37）。

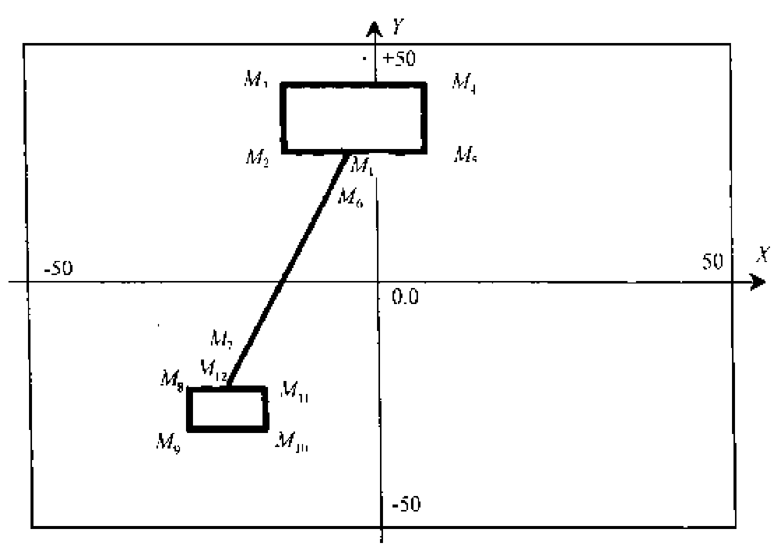


图 5-37 矢量 Mx 和 My 图解

animat.m file

```
% graphical simulation of the gantry
function [sys,x0,str,ts] = animat(t, x, u, flag,Te)
global gantry4
switch flag,
case 0    % Initialisation stage

% figure initialisation
animinit('Gantry animation');
gantry4 = findobj('Type','figure','Name','gantry animation');
axis([-50 50 -50 50])
hold on

% Plotting fixed image ( ground + gantry )
 plot([-50 50],[-40 -40],'blue',...
    [-50:50;-50:50],[-45 -40],'blue','LineWidth',1)
 plot([-49 -49 -45 -45],[-39 35 35 -39],'black',...
    [49 49 45 45],[-39 35 35 -39],'black','LineWidth',4)
 plot([-40 40],[35 35],'black',...
    [-50:50;-50:50],[35 37],'black','LineWidth',5)

 % Initialisation of the mobile plotting
 %( truck + suspended mass )
 Mx = [0 -5 -5 5 5 0 0 -2 -2 2 2 0];
 My = [30 30 34 34 30 30 -20 -20 -22 -22 -20 -20];
 % Plotting the mobile image and returning a pointer on the
 % graphic object

 hndl = plot(Mx,My,'blue','erasemode','xor','linewidth',2);
 set(gca,'UserData',hndl);

 % Calling Initialisation system
 [sys,x0,str,ts] = Initialisation(Te);

 case 2  % acquisition of the inputs-outputs vector for
        % up-dating display

 % If the figure exists, pointer recuperation
 if any(get(0,'Children')==gantry4),
 if strcmp(get(gantry4,'Name'),'Gantry animation'),
 set(0,'currentfigure',gantry4);
 hndl = get(gca,'UserData');

 % reading the Inputs-Outputs and vectors Mx and My
 % calculation
        Cx = 10*u(1);
        [tetay,tetax] = pol2cart(u(2),50);
        Px = Cx+tetax;
        Mx = [Cx Cx-5 Cx-5 Cx+5 Cx+5 Cx Px Px-2 Px-2 ...
              Px+2 Px+2 Px];
        My = [30 30 34 34 30 30 30-tetay 30-tetay ...
```

```
            28-tetay 28-tetay 30-tetay 30-tetay];
        % plot up-dating
        set(hndl,'Xdata',Mx,'Ydata',My);
        drawnow
      end
    end
    sys = [];

case {1,3,4,9} % non-used stages
    sys = [];
otherwise
    error(['Unhandle flag = ',num2str(flag)])
end

% Initialisation function
  function [sys,x0,str,ts] = Initialisation(Te)
  sizes = simsizes;           % structure of type simsizes
  sizes.NumContStates = 0; % continuous states number=0
  sizes.NumDiscStates = 0; % discrete states number=0
  sizes.NumOutputs = 0;       % outputs number = 0
  sizes.NumInputs = 2;        % inputs number = 2
  sizes.DirFeedthrough = 0;      % non using of D
  sizes.NumSampleTimes = 1;      % lines number of ts
  sys = simsizes(sizes);         % vector up-dating
  x0 = [];                     % non specified initial state
  str = [];                    % empty chain
  ts = [Te 0];                 % sampling period = Te
                               % (passed as parameter)
```

这样得到的制作图形如图 5-38 所示。

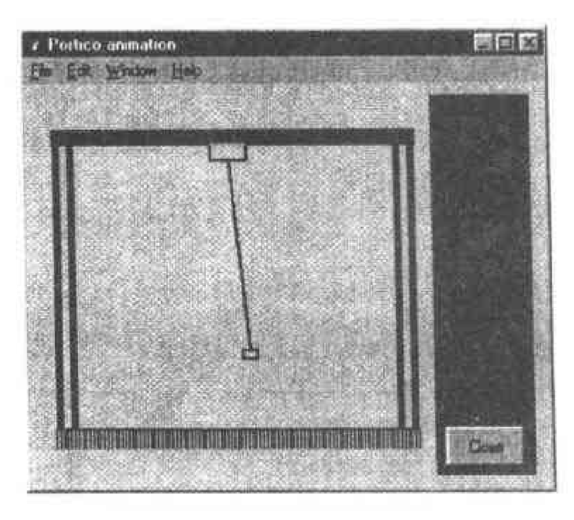


图 5-38　得到的制作图形

5.7　吊架的模糊控制

使用 2 个模糊独立的控制器控制吊架：

- 名为 fuzz_x 的用于位置 $x(t)$ 的第 1 控制器：

● 名为 fuzzy_teta 的用于角 $\theta(t)$ 的第 2 控制器。

文件 def_reg.m 可以用来表示隶属函数，每个控制器定义 2 个输入和 1 个输出以及干扰规则。对每个变量使用 3 个高斯法则。

def_reg.m file

```
% Position fuzzy regulator definition
sys_fuzzy  =  readfis('fuzzy_x');

% membership functions drawing
figure(1)
plotmf(sys_fuzzy,'input',1)
xlabel('truck position (m)')
ylabel('membership degree')
title('fuzzy regulator de position')
figure(2)
plotmf(sys_fuzzy,'input',2)
xlabel('truck displacement variation (m)')
ylabel('membership degree')
title('position fuzzy regulator')
figure(3)
plotmf(sys_fuzzy,'output',1)
xlabel('force to be applied (N)')
ylabel('membership degree')
title(' position fuzzy regulator')
```

◆ 位置 $x(t)$ 的隶属函数（见图 5-39）

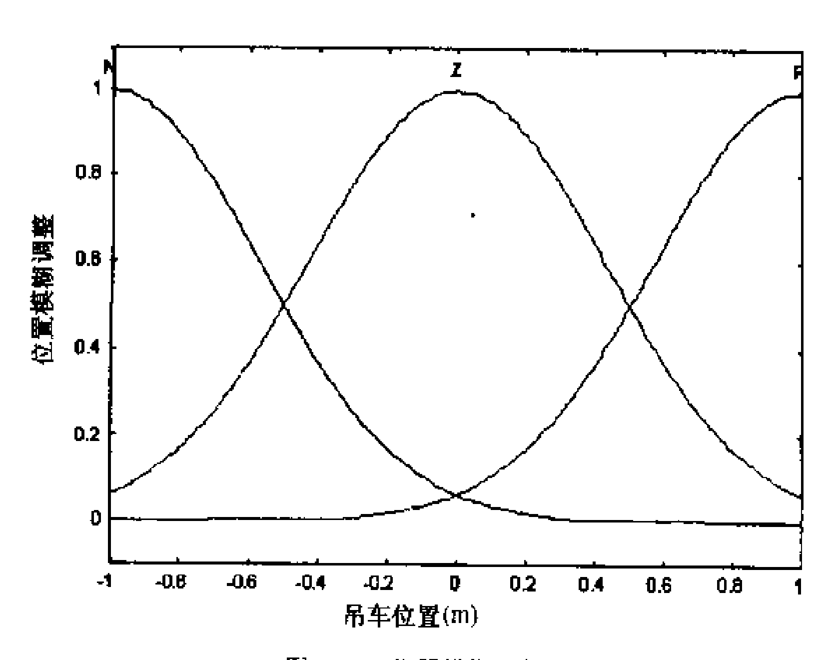


图 5-39 位置模糊调整

◆ 位置微分 $\dot{x}(t)$ 的隶属函数（见图 5-40）

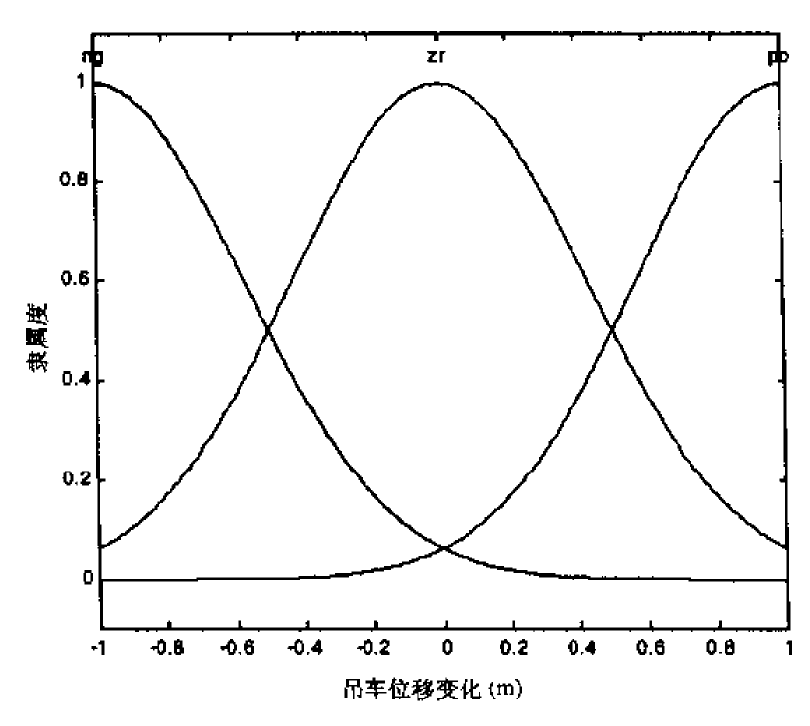


图 5-40　位置模糊调整

◆　输出 $f(t)$ 的隶属函数（见图 5-41）

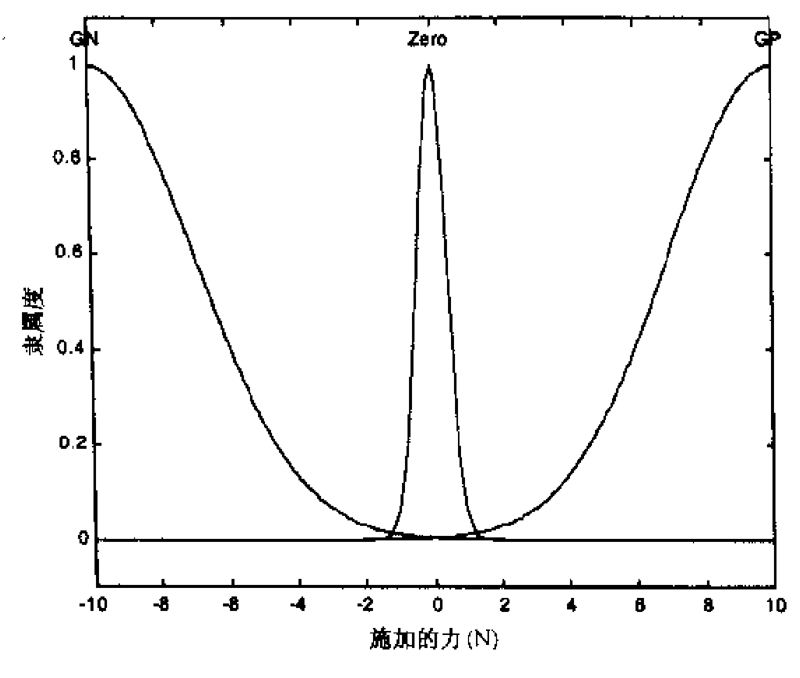


图 5-41　位置模糊调整

代表 2 个输入和 3 个规则的控制器由 9 个模糊规则描述。它们在 verbal 模式下编辑（见图 5-42）。

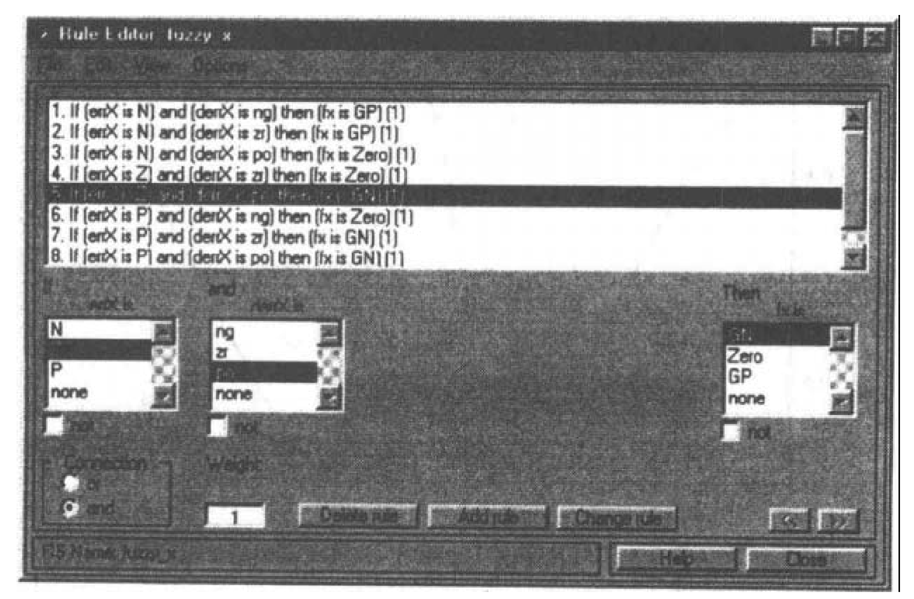


图 5-42　在 verbal 模式下的编辑

def_reg.m(follow)

```
% fuzzy regulator definition for the oscillations
sys_fuzzy = readfis('fuzzy_teta');

% Drawing of the 2 inputs membership functions
figure(4)
% input n° 1
plotmf(sys_fuzzy,'input',1)
xlabel('suspended mass' angle (rad)')
ylabel('membership degree')
title('suppression of oscillations')

figure(5)
% entrée n° 2
plotmf(sys_fuzzy,'input',2)
xlabel('angle variations (rad)')
ylabel('membership degree')
title('suppression of oscillations')

% Drawing the output membership functions
figure(6)
plotmf(sys_fuzzy,'output',1)
title('cancellation of oscillations')
xlabel('force to be applied (N)')
ylabel('membership degree')
```

◆　悬挂物角度变化的隶属函数（见图 5-43）

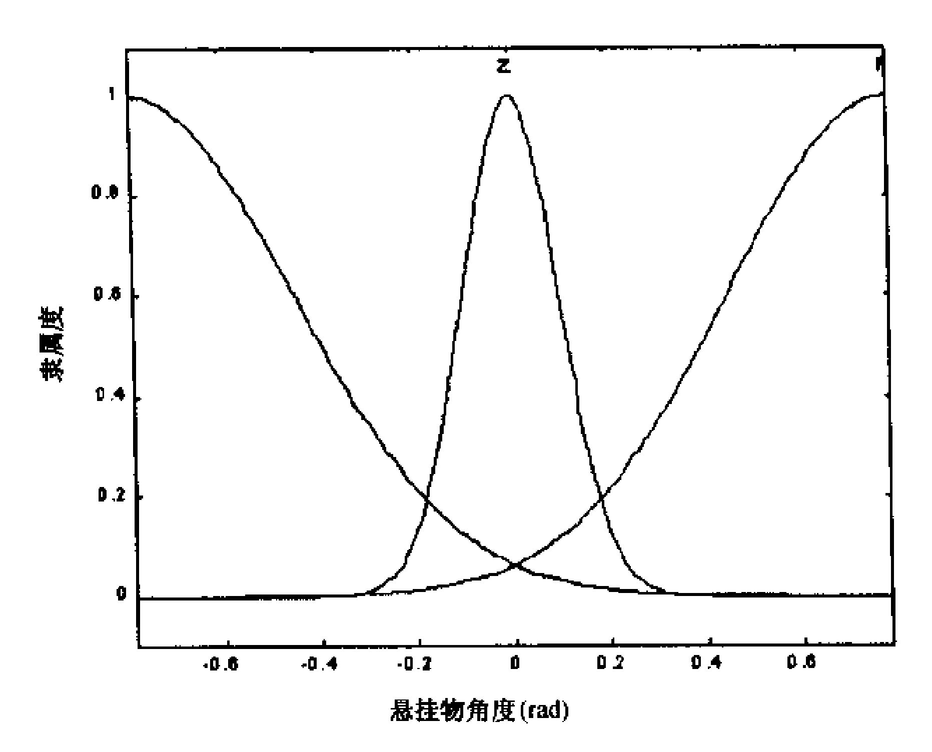


图 5-43　悬挂物角度摆动的抑制

◆　悬挂物角度的微分 $\dot{\theta}'(t)$ 的隶属函数（见图 5-44）

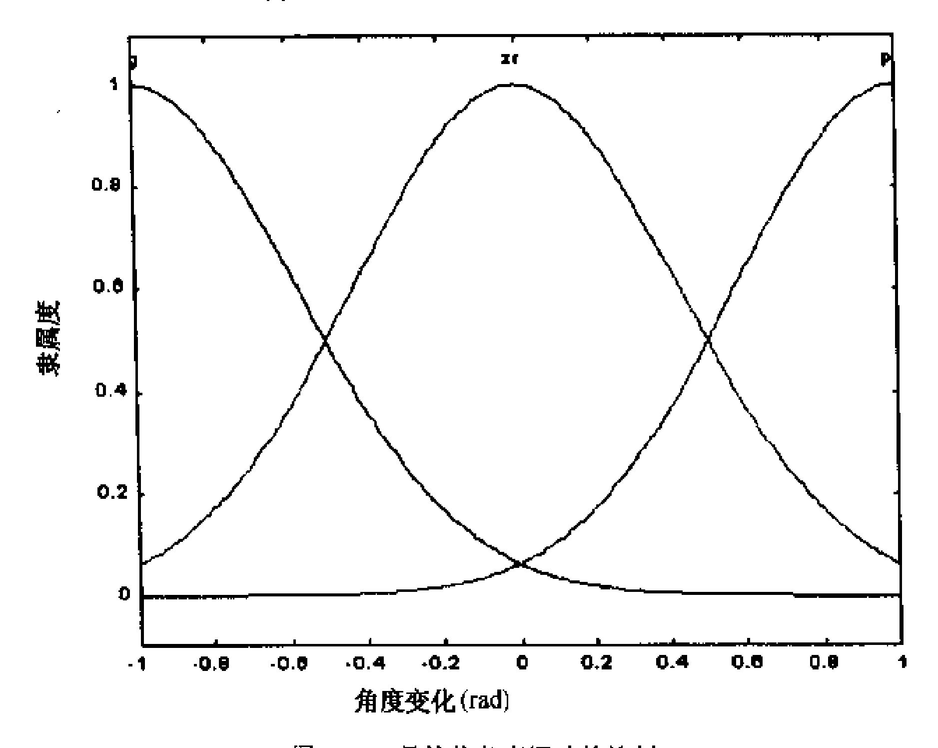


图 5-44　悬挂物角度摆动的抑制

◆　力 $f(t)$ 的隶属函数（见图 5-45）

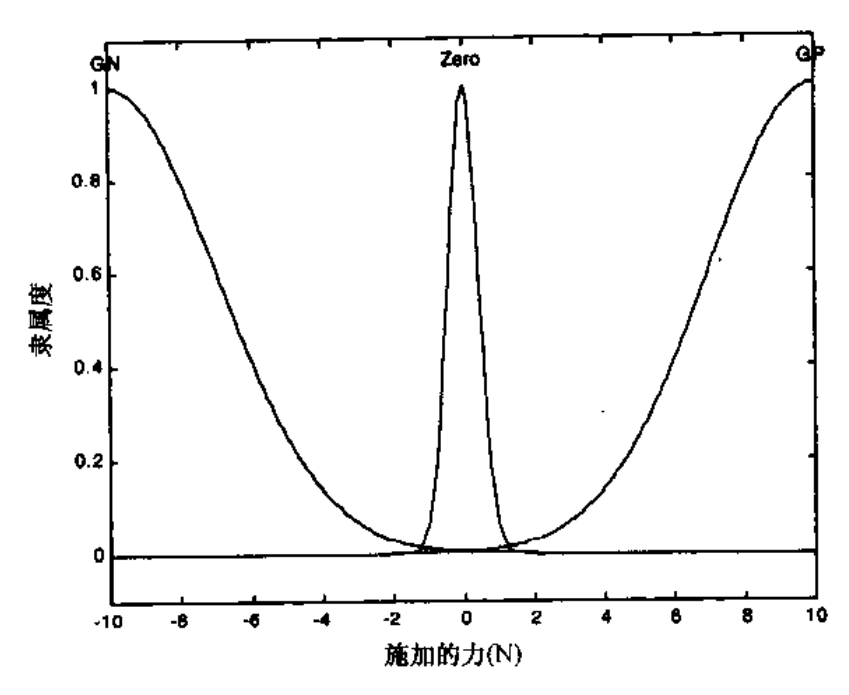


图 5-45　摆动的消失

◆　在 verbal 模式下编辑的 fuzzy_teta 调节器的模糊规则（见图 5-46）

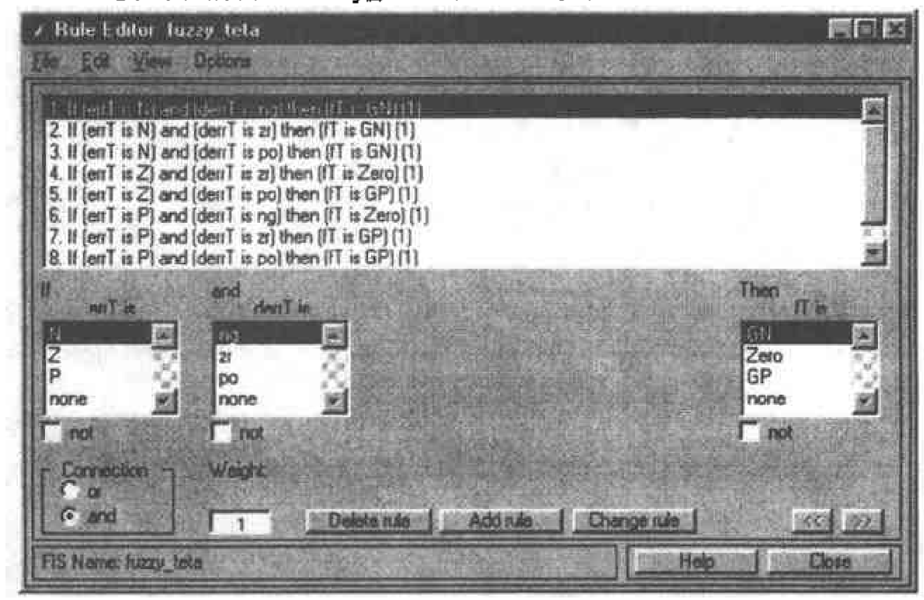


图 5-46　在 verbal 模式下编辑 fuzzy_teta

我们使用在块 Fcn 编辑下的吊架非线性数学模型进行仿真。为实现离散控制而采用零阶保持。由于模糊矩阵 fuzzy_x，变量 x (t)位置以及它的微分被加于控制器上。由于模糊矩阵 fuzzy_teta，变量θ (t)、悬挂物角度以及它的微分被加于相应的控制器上。然后将 2 个控制器的输出相加以控制位置和角度。在角度输出控制器中增加一个参数以调节这个变量。

◆　SIMULINK 模型的模糊控制（见图 5-47）

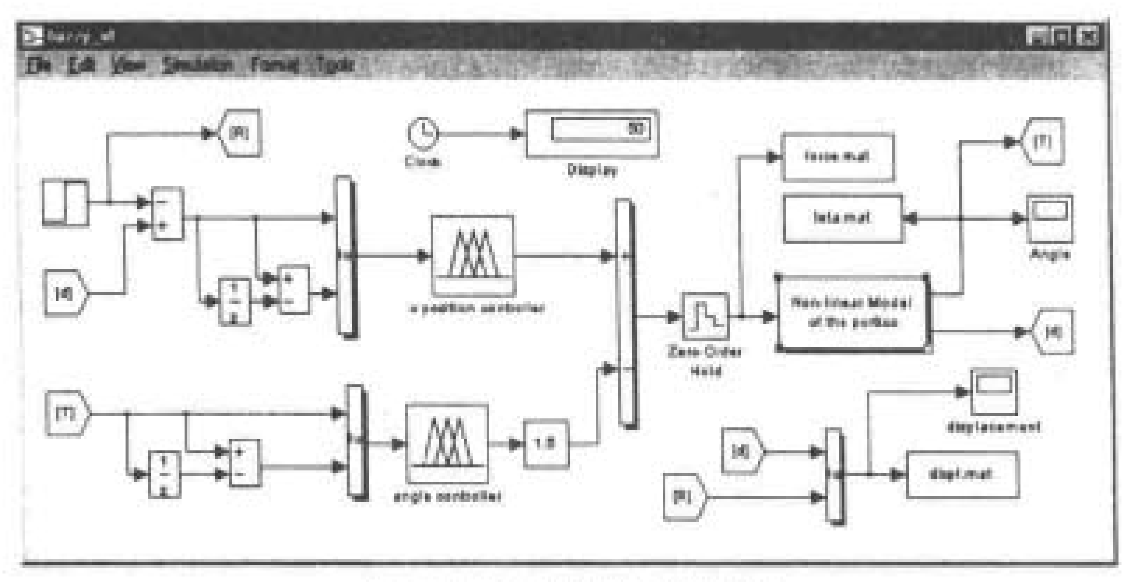


图 5-47　SIMULINK 模型的模糊控制

文件 es_portik.m 可检验位置 x 阶跃 1 m 的控制系统响应。

es_portik.m file

```
% gantry control by fuzzy regulator
% reading the input/output and display results
close all
load force.mat
f = f(2,:);            % applied force to the process
load teta.mat
teta = teta(2,:);      % angle of the cable
teta = teta*180/pi; % angle in degrees
load displ.mat
t = x(1,:);
d = x(2,:);        % cart displacement
r = x(3,:);        % displacement set-point
% plots
figure(1)
plot(t,r,':')
hold on, plot(t,d)
title('set-point and truck displacement in m')
xlabel('discrete time')
figure(2)
stairs(t,f), hold on
title('applied force to the portico (N)')
xlabel('discrete time')
figure(3)
stairs(t,teta), hold on
title('suspended mass angle in deg')
xlabel('discrete time')
```

设定点的值在大约 20 s 后达到。

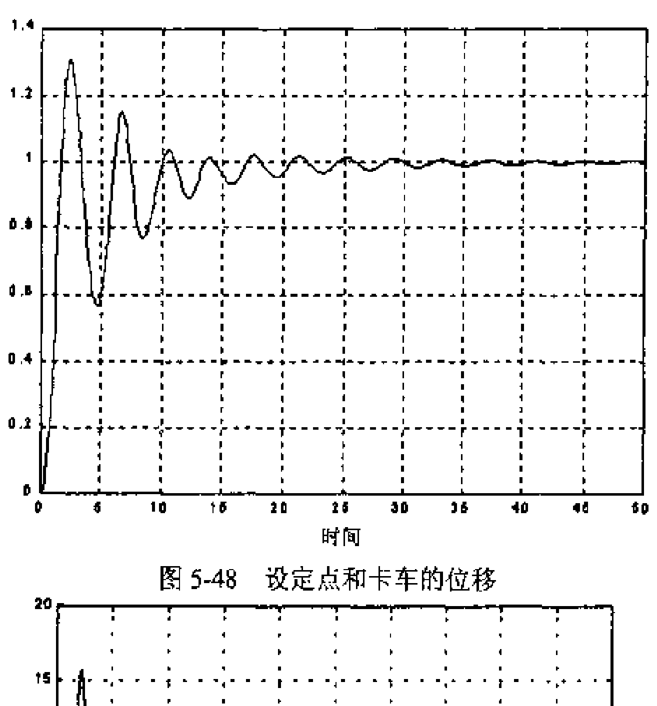


图 5-48　设定点和卡车的位移

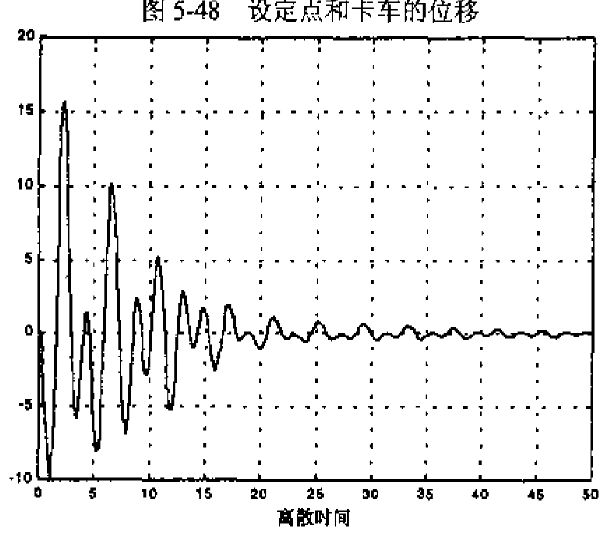


图 5-49　悬挂物的角度

吊车在理想位置附近的振荡对悬挂物有影响，使它在位置 x 已稳定后仍保持振荡。加在吊车上的力在许多情况下反转自身，这是因为每个控制器以相反方式作用。

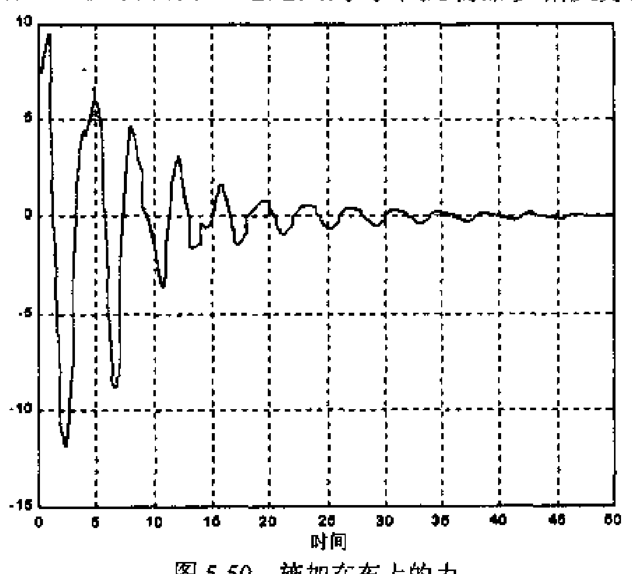


图 5-50　施加在车上的力

5.8 RST 和 LQI 控制器

5.8.1 吊架的离散模型

如前所述，吊架被如下微分方程定义：

$$\ddot{y}(t) = -g\frac{M+m}{Ml}y(t) - \frac{1}{Ml}f(t)$$

$$\ddot{x}(t) = -\frac{mg}{M}y(t) - \frac{1}{M}f(t)$$

力 $y(t)$与悬挂物角度相联系的传递函数可以从第 1 式中计算出来。于是，得到

$$\frac{Y(p)}{F(p)} = -\frac{1}{Mlp^2 + g(M+m)}$$

通过从上面得出的方程来代替 $Y(p)$，得到如下将力与位置变量联系起来的传递函数：

$$\frac{X(p)}{F(p)} = -\frac{lp^2 + g}{Mlp^4 + g(M+m)p^2}$$

吊架可被具有 1 个输入、2 个输出的多变量模型来描述。

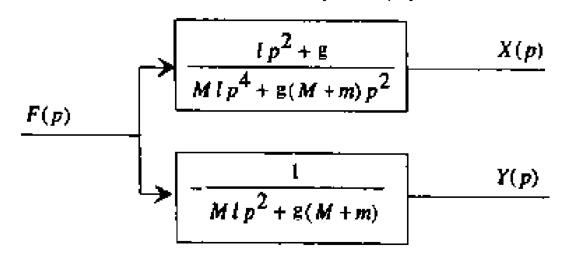


图 5-51 吊架的多变量模型

文件 mod_gantry.m 可得到上面模型中用 0.1 s 采样周期而得到的离散传递函数以及插入 LQI 和 RST 控制器的矩阵。

mod_gantry.m

```
function [Axet, Bxet, bx, Ayet, Byet, by] = mod_portico
% process parameters

M = 10; m = 5; g = 9.81; l = 1;

% parallel analogical model
% analogical step responses of the model
sys_Xc = tf([l 0 g],[M*l  0 g*(M+m) 0 0]);
sys_Yc = tf(-1,[M*l 0 g*(M+m)]);

% discrete model
Ts = 0.1; % sampling period
sys_Xd = c2d(sys_Xc,Ts,'zoh');
sys_Yd = c2d(sys_Yc,Ts,'zoh');

% numerator and denominator of the transfer functions
[numX,denX] = tfdata(sys_Xd,'v');
```

```
[numY,denY] = tfdata(sys_Yd,'v');

% portico's model parameters
% x displacement model

Axet = -denX(2:length(denX));
Bxet = numX(3:length(numX));
bx = numX(2);
Ayet = -denY(2:length(denY));
Byet = numY(3:length(numY));
by = numY(2);
```

回忆并不需要在命令规则中加入积分的预测模型：

$$y(t+1)=A^*(z)y(t)+B^*(z)u(t-1)+Bu(t)$$

在本吊架中，有如下4位小数的值：

$$A^*(z)=\begin{bmatrix}3.8546-5.7093z^{-1}+3.8546z^{-2}-z^{-3} & 0\\ 0 & 1.8546-z^{-1}\end{bmatrix}=\begin{bmatrix}A_x^* & 0\\ 0 & A_y^*\end{bmatrix}$$

可如下表示：

$$A^*(z)=\begin{bmatrix}3.8546 & 0\\ 0 & 1.8546\end{bmatrix}+\begin{bmatrix}-5.7093 & 0\\ 0 & -1\end{bmatrix}z^{-1}+\begin{bmatrix}3.8546 & 0\\ 0 & 0\end{bmatrix}z^{-2}+\begin{bmatrix}-1 & 0\\ 0 & 0\end{bmatrix}z^{-3}$$

$$=A_0^*+A_1^*z^{-1}+A_2^*z^{-2}+A_3^*z^{-3}$$

以及

$$B^*(z)=\begin{bmatrix}-0.00044951-0.00044951z^{-1}+0.00049797z^{-2}\\ -0.0004939\end{bmatrix}=\begin{bmatrix}B_x^*\\ B_y^*\end{bmatrix}$$

$$=\begin{bmatrix}-0.00044951\\ -0.0004939\end{bmatrix}+\begin{bmatrix}-0.00044951\\ 0\end{bmatrix}z^{-1}+\begin{bmatrix}0.00049797\\ 0\end{bmatrix}z^{-2}$$

$$=B_0^*+B_1^*z^{-1}+B_2^*z^{-2}$$

$$B=\begin{bmatrix}0.00049797\\ -0.0004939\end{bmatrix}=\begin{bmatrix}b_x\\ b_y\end{bmatrix}$$

文件 mod_gantry.m 可计算 Axet、Bxet、bx、Ayet、Byet 和 by 矩阵。下面来检查这些矩阵的值。

```
>> [Axet ,Bxet,bx,Ayet,Byet,by]=mod_gantry
Axet=
   3.8546   -5.7093    3.8546   -1.0000
Bxet=
0e-003*
-0.4495    -0.4495     0.4980
bx=
   4.9797e-004

Ayet=
-1.0000
Byet=
   -4.9390e-004
```

```
by=
   -4.9390e-004
```

5.8.2 RST 控制规则

5.8.2.1 吊车位置的 RST 单变量控制

在如下部分，建议只调节吊车位置 x。预测器模型为

$$y(t+1)=A^*(z)y(t)+B^*(z)u(t-1)+Bu(t)$$

假设

$$A^*(z)=3.8546-5.7093z^{-1}+3.8546z^{-2}-z^{-3}$$

$$B^*(z)=-0.00044951-0.00044951z^{-1}+0.00049797z^{-2}$$

以及

$$B=0.00049797$$

文件 rst_x.m 实现了位置 x 的单变量 RST 控制。选择具有阻尼系数 $\xi=\frac{\sqrt{2}}{2}$ 的二阶动态追踪。

rst_x.m

```
close all, clear all

% dynamic of regulation, first order of 0.8 pole
P = 0.8;

% discrete model matrices
[Axet, Bxet, bx, Ayet, Byet, by] = mod_portico;
B = bx;

% displacement and angle set-points generation
x_c = [-ones(1,100) ones(1,200) -ones(1,150) ];

% position set-point reference model
% second order, dzeta = 0.7071
% second order référence model
dzeta = sqrt(2)/2;
w0T = 0.05*pi;
alfa1 = -2*exp(-dzeta*w0T)*cos(w0T*sqrt(1-dzeta^2));
alfa2 = exp(-2*dzeta*w0T);
rx(1:2) = x_c(1:2);

for i = 3:length(x_c)
  rx(i) = (1+alfa1+alfa2)*x_c(i)-alfa1*rx(i-1)-alfa2*rx(i-2);
end

% initialisation
f = zeros(size(x_c));
x = rx;
y = zeros(size(x_c));
```

```
for k = 5:length(f)-1
  % displacement response
  X = [x(k-1) x(k-2) x(k-3) x(k-4)];
  F = [f(k-1) f(k-2) f(k-3) f(k-4)];
  x(k) = X*Axet'+[bx Bxet]*F';

  % angle response
  Y = [y(k-1) y(k-2)];
  y(k) = Y*Ayet'+F(1:2)*[by Byet]';

  Phi = rx(k+1)-P*rx(k)+[(P-Axet(1)) -Axet(2:4)]*...
      [x(k) x(k-1) x(k-2) x(k-3)]';
  Phi = Phi-Bxet*[f(k-1) f(k-2) f(k-3)]';
  f(k) = inv(B'*B)*B'*Phi;

  % position disturbance
  x(k) = x(k) - 0.1*(k == 210);
end
figure(1)
h = plot(x,'r');
set(h,'LineWidth',2);
hold on
plot(rx,'b:')
plot(x_c,'g');
axis([0 400 -1.2 1.2])
title('RST control of the position')
text(110,0,'\leftarrow position of the cart',...
  'HorizontalAlignment','left')
text(300,-0.8,'reference position\rightarrow',...
  'HorizontalAlignment','right')
xlabel('samples')

figure(2)
h = plot(y);
set(h,'LineWidth',2);
title('RST control of the position')
xlabel('samples')
text(230,-8,'angle in rad \rightarrow',...
  'HorizontalAlignment','right')

figure(3), stairs(f)
xlabel('samples')
title('RST control of the position')
gtext('applied force in N')
```

在 t=210 时，产生一个幅度为 0.1 m 的扰动，根据特定的动态，它将被抑制（见图 5-52）。有了这种类型的控制，调节和跟踪目标是独立的。

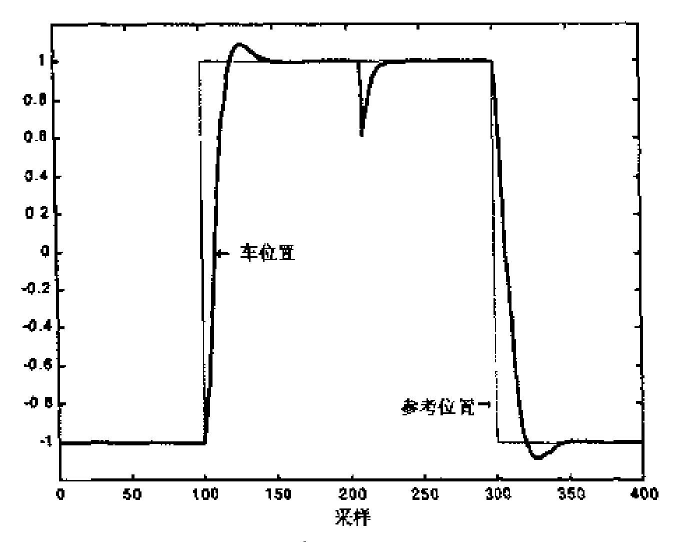


图 5-52　位置的 RST 控制

因为达不到对任何悬挂物的偏离角度调节，所以角度对位置干扰的影响有反应（见图 5-53）。

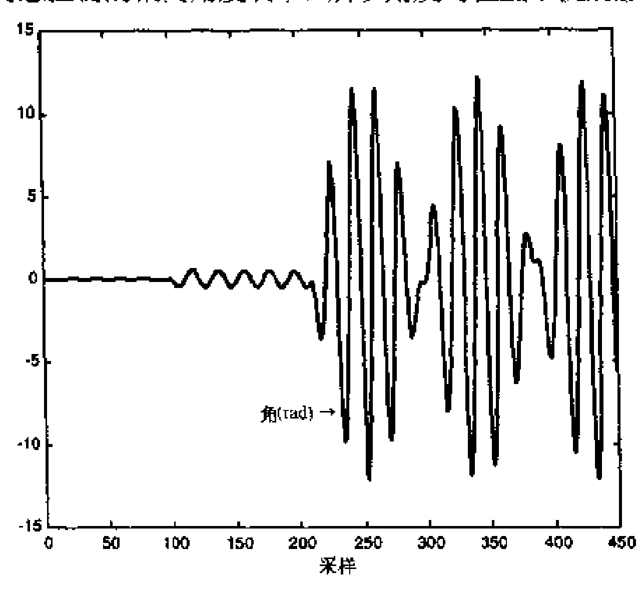


图 5-53　位置的 RST 控制

图 5.54 所示为在干扰前后施加的力。

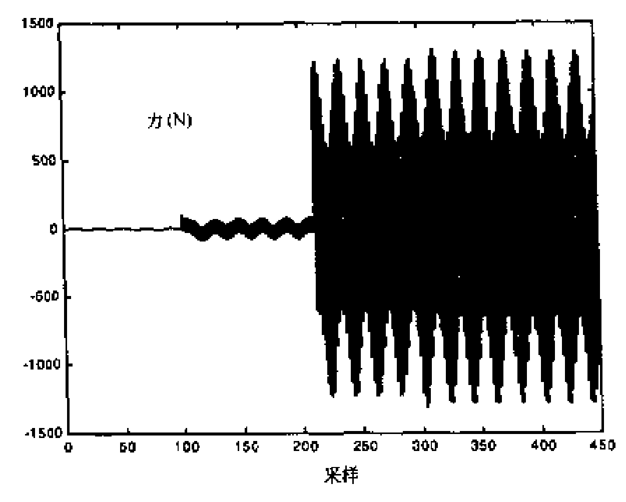


图 5-54　位置的 RST 控制

5.8.2.2 吊架的多变量 RST 控制

吊架对车的位置和悬挂物的偏离角度的多变量控制通过文件 rst_xy.m 实现。不管是位置还是角度偏离，干扰的动态都由如下多项式确定：

$$Px(z) = 1 - pxz^{-1} = 1 - 0.9z^{-1}$$

$$Py(z) = 1 - py_1 z^{-1} = 1 - 0.5z^{-1}$$

位置跟踪的动态是一个具有阻尼系数ξ =1 的二阶过程。由于我们试图根据这个动态达到设定点，同时消除悬挂物的偏离角度，于是零点角度设定点没有被过滤掉，它直接组成了参考信号。

rst_xy.m file

```
% dynamic regulation matrix
% first order dynamic for the displacement
px1 = 0.9;
% first order dynamic for the angle
%py1 = 0.2;
py1 = 0.8;
% regulation matrix
P = [px1 0; 0 py1];
% calling the mod_gantry file for matrices calculation
[Axet, Bxet, bx, Ayet, Byet, by] = mod_gantry;
% multivariable control matrices
A0et = [Axet(1)    0
        0          Ayet(1)];
A1et = [Axet(2)    0
        0          Ayet(2)];
A2et = [Axet(3)    0
        0          0];
A3et = [Axet(4)    0
        0          0];

B0et = [Bxet(1); Byet(1)];
B1et = [Bxet(2); 0];
B2et = [Bxet(3); 0];

B = [bx; by];

% position and angle set-points generation
x_c = [-ones(1,100) ones(1,350)];

% reference models for the position
dzeta = 1;
w0T = 0.05*pi;
alfa1 = -2*exp(-dzeta*w0T)*cos(w0T*sqrt(1-dzeta^2));
alfa2 = exp(-2*dzeta*w0T);
rx(1:2) = x_c(1:2);

for i = 3:length(x_c)
  rx(i) = (1+alfa1+alfa2)*x_c(i)-alfa1*rx(i-1)-alfa2*rx(i-2);
```

```
end

y_c = zeros(size(x_c));

% reference signals
xy_c = [rx; y_c];

% initialisation
f = zeros(size(x_c));
x = rx;
y = zeros(size(f));
xy = [x; y];

for k = 5:length(f)-1

  % displacement response
  X = [x(k-1) x(k-2) x(k-3) x(k-4)];
  F = [f(k-1) f(k-2) f(k-3) f(k-4)];
  x(k) = X*Axet'+[bx Bxet]*F';

  % angle response
  Y = [y(k-1) y(k-2)];
  y(k) = Y*Ayet'+F(1:2)*[by Byet]';

  % xy measured vector
  xy(:,k) = [x(k);y(k)];
  Phi = xy_c(:,k+1)-P*xy_c(:,k)+(P-A0et)*xy(:,k)-A1et*...
        xy(:,k-1)-A2et*xy(:,k-2)-A3et*xy(:,k-3);
  Phi = Phi-B0et*f(k-1)-B1et*f(k-2)-B2et*f(k-3);
  f(k) = inv(B'*B)*B'*Phi;


end

% plotting of the cart position
figure(1)
h = plot(x,'r');
set(h,'LineWidth',2)
hold on
plot(rx,'b:')
plot(x_c,'g')
axis([0 400 -1.2 1.5])
title('RST multivariable control of the gantry')
gtext('\leftarrow position of the cart')
text(100,-0.5,'set-point\rightarrow',...
    'HorizontalAlignment','right')
text(115,0,'reference\rightarrow ',...
    'HorizontalAlignment','right')

xlabel('samples')
```

```
% plotting of the angle deviation of the suspended mass
figure(2)
h = plot(y);
set(h,'LineWidth',2)
hold on
plot(y_c)
title('RST multivariable control of the gantry')
xlabel('samples')
gtext('\leftarrow deviation angle of the mass (degrees)');

% plotting the applied force
figure(3)
stairs(f)
xlabel('samples')
title('RST multivariable control of the gantry')
gtext('applied force in N')
```

图 5-55 说明参考信号的到达以及角度调节对位置曲线的影响。

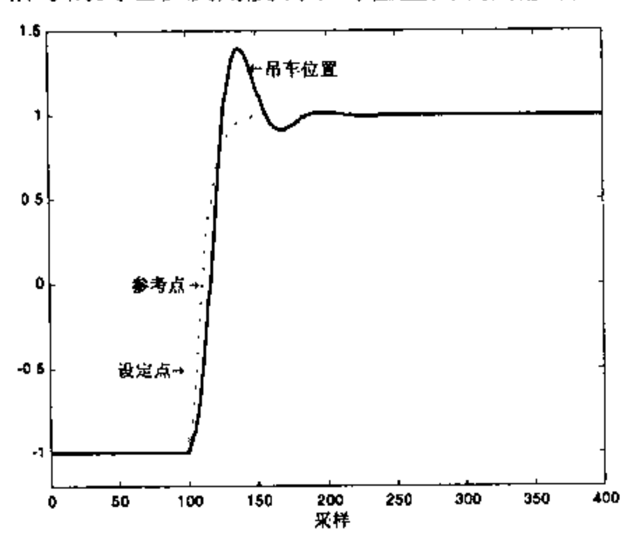


图 5-55　吊车的 RST 多变量控制

图 5-56 显示的是角度对过滤位置阶跃的响应。

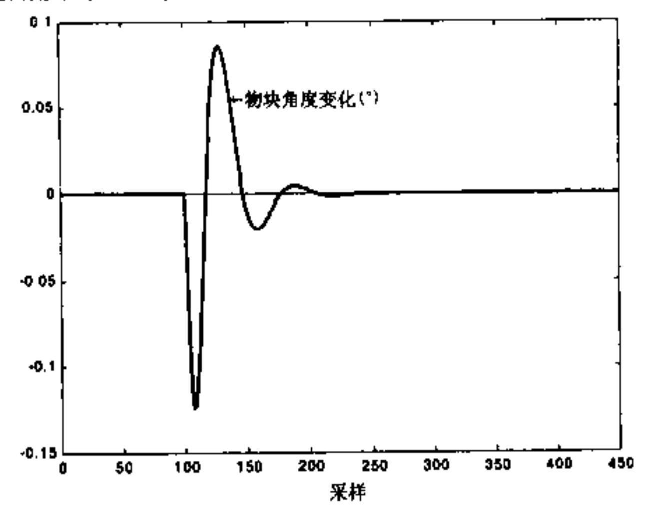


图 5-56　吊车的 RST 多变量控制

为保持车处于稳定位置，力在-30 N~+30 N 之间变化。

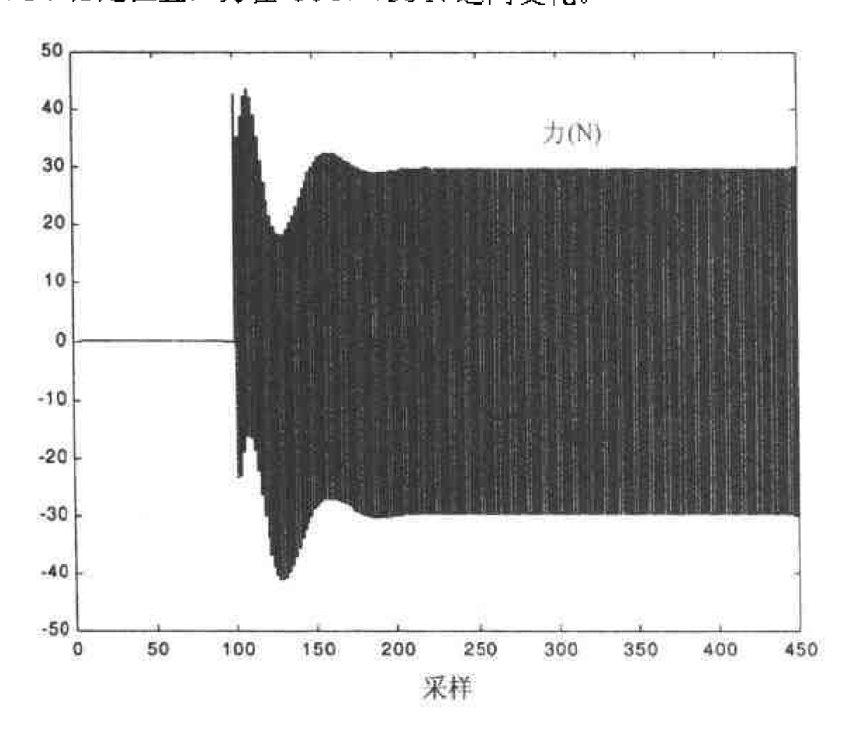


图 5-57　吊车的 RST 多变量控制

物块的振荡可能对位置超调有影响，因此有必要减少角度动态，使其对位置影响更小。如果选一个由调节多项式 $Py(z)=1-py_1z^{-1}=1-0.8z^{-1}$ 确定的慢角速度，就得到在位置上有振荡和一个较大的时间响应的输出。

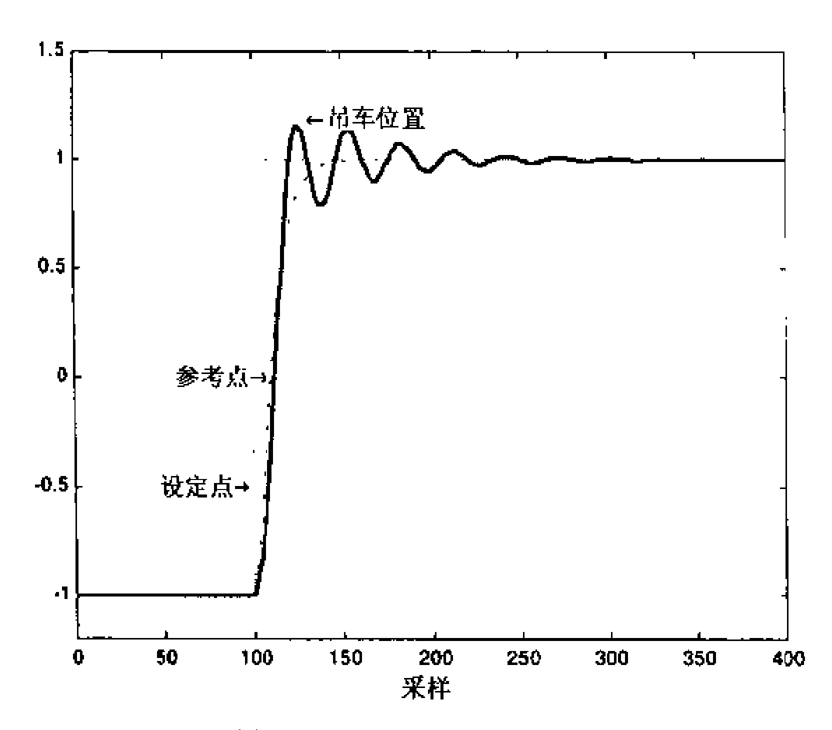


图 5-58　吊车的 RST 多变量控制

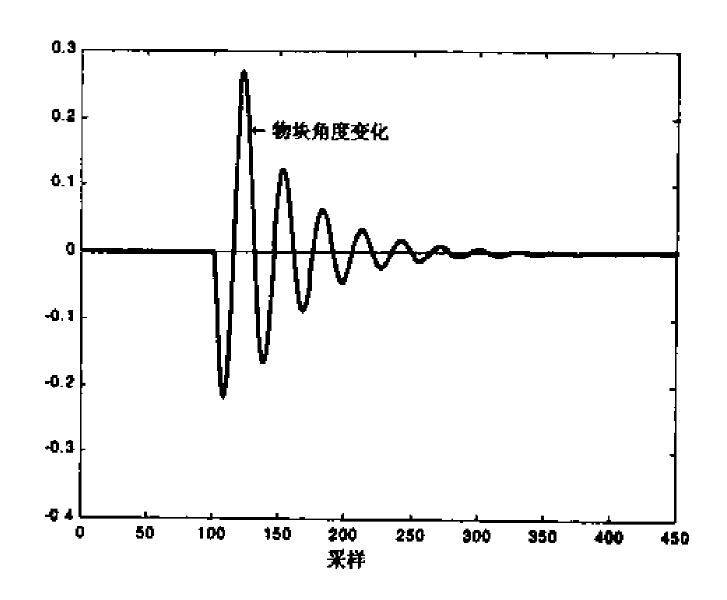


图 5-59　吊车的 RST 多变量控制

5.8.3　吊车位置的 LQI 单变量控制

在瞬间作用的力 $f(t)$ 有如下表达式：

$$f(t)=\frac{b_1}{b_1^{\,2}+R}\left[r(t+1)-A^*(z)y(t)-B^*(z)u(t-1)\right]$$

文件 lqi_x.m 执行这个控制规则。根据具有阻尼系数 $\xi=\frac{\sqrt{2}}{2}$ 的二阶动态来达到阶跃设定点。在平衡系数 $R=10^{-8}$ 下，设定点的跟踪在 $R=0$ 是准最优的。相反，外加力的变化明显是最小的。

lqi_x.m

```
[Axet, Bxet, bx, Ayet, Byet, by] = mod_portico;
% position and angle set-points generation
x_c = [-ones(1,100) ones(1,200) -ones(1,150)];
% reference model for the position
dzeta = sqrt(2)/2;
w0T = 0.05*pi;
alfa1 = -2*exp(-dzeta*w0T)*cos(w0T*sqrt(1-dzeta^2));
alfa2 = exp(-2*dzeta*w0T);
rx(1:2) = x_c(1:2);
for i = 3:length(x_c)
  rx(i) = (1+alfa1+alfa2)*x_c(i)-alfa1*rx(i-1)-alfa2*rx(i-2);
end
% R balancing coefficient of the force
%R = 1e-8;
%R=0
R=1e-6
% coefficient eta
eta = bx/(bx^2+R);
% initialisation
f = zeros(size(x_c));
```

```
x = x_c;
y = zeros(size(x));
for k = 5:length(x_c)-1
  % displacement response
  X = [x(k-1) x(k-2) x(k-3) x(k-4)];
  F = [f(k-1) f(k-2) f(k-3) f(k-4)];
  x(k) = Axet*X'+[bx Bxet]*F';
  % angle response
  Y = [y(k-1) y(k-2)];
  F = [f(k-1) f(k-2)];
  y(k) = Ayet*Y'+[by Byet]*F';
  Phi = rx(k+1)-Axet*[x(k) x(k-1) x(k-2) x(k-3)]';
  Phi = Phi-Bxet*[f(k-1) f(k-2) f(k-3)]';
  f(k) = eta*Phi;
end
figure(1)
h = plot(x);
set(h,'LineWidth',2), hold on
plot(x_c,':')
plot(rx,'-.')
axis([0 400 -1.2 1.2])
title('LQI control of the cart position')
ylabel('cart position (meters)')
xlabel('samples')
figure(2)
y = y*180/pi;
plot(y)
title('LQI control of the cart position')
ylabel('suspended mass deviation angle (degrees)')
xlabel('samples')
figure(3), stairs(f)
xlabel('samples')
title('LQI control of the cart position')
ylabel('applied force in N')
```

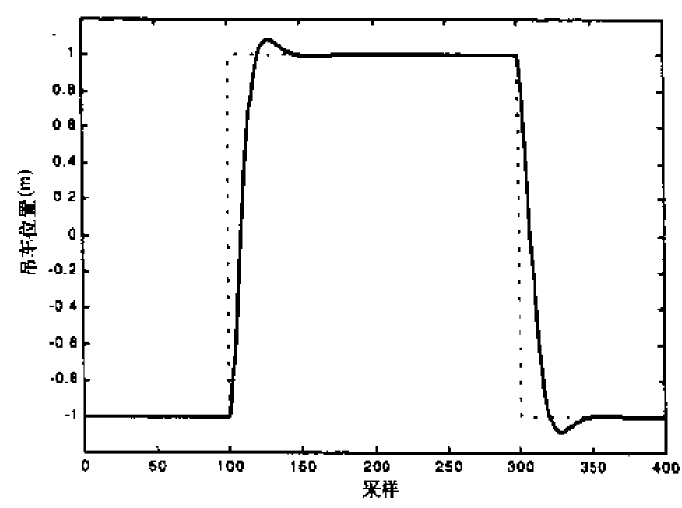


图 5-60　吊车位置的 LQI 控制

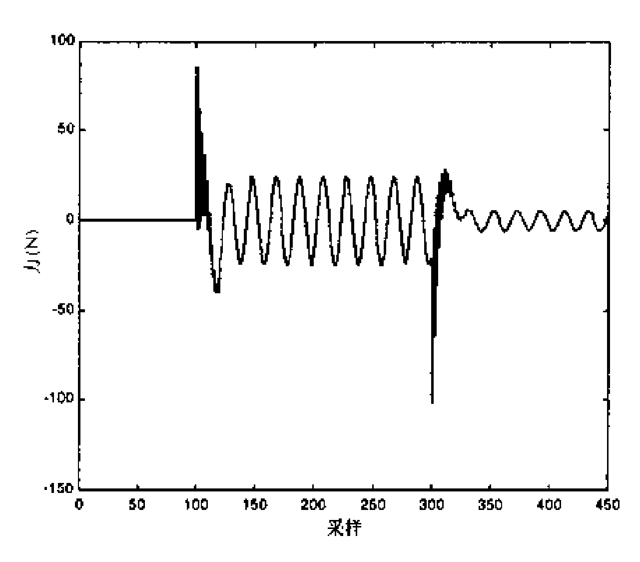


图 5-61　吊车位置的 LQI 控制

当 R=0 时，跟踪与前面的情况是准一致的，对悬挂物偏离角度有同样的效果。相反，所加力的变化是最大的。

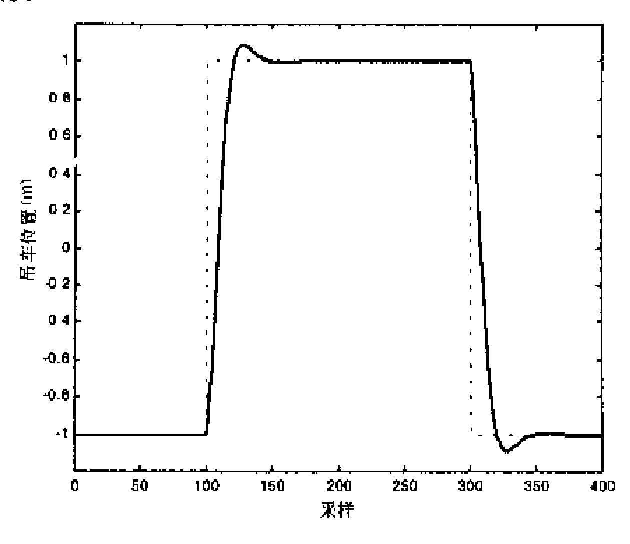


图 5-62　吊车位置的 LQI 控制

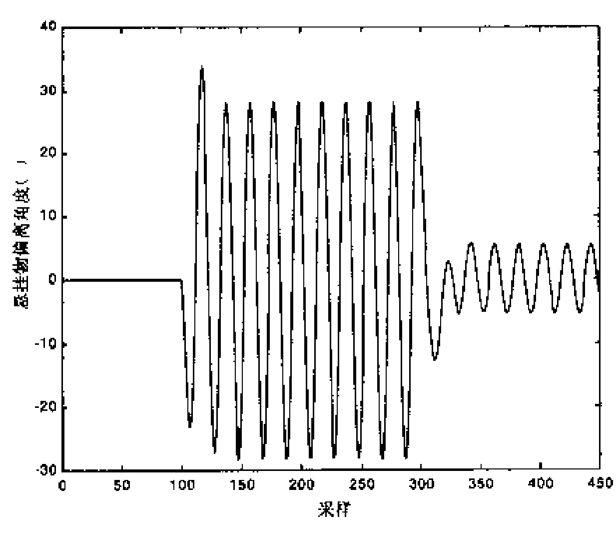


图 5-63　吊车位置的 LQI 控制

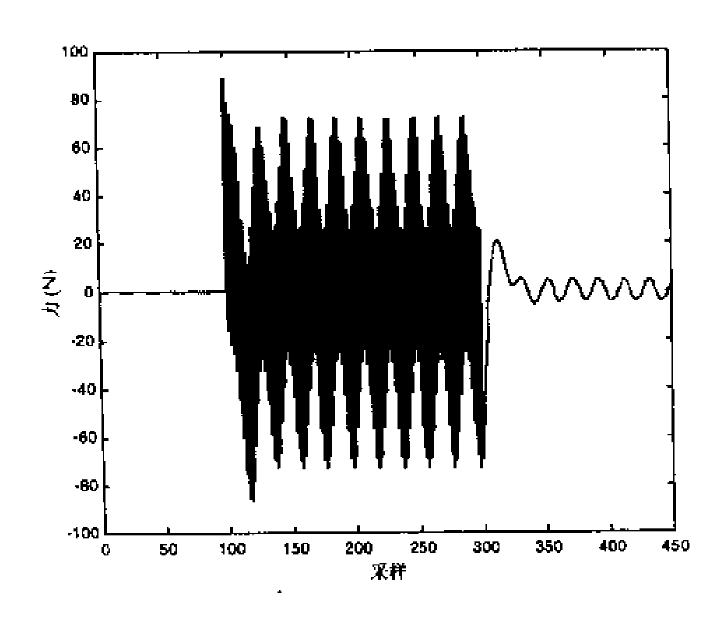


图 5-64　吊车位置的 LQI 控制

R 系数值增加，超过10^{-8}，导致在稳定状态下在参考点附近输出信号发生振荡。图 5-65 显示的是 R=10^{-6} 时的振荡。

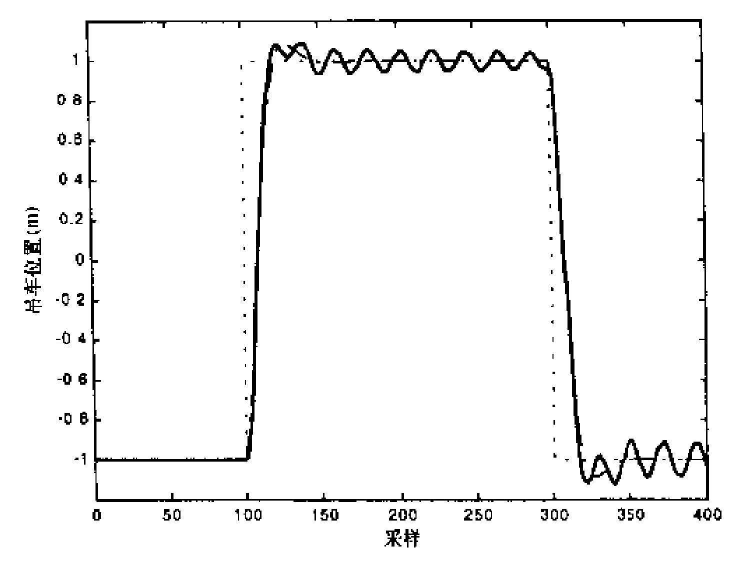


图 5-65　吊车位置的 LQI 控制

应用6 免 提 电 话

设想移动电话放在汽车控制面板这一情况。发动机产生的噪声 p 经一倍增系数叠加在组成有用信号 x 的驾驶员声音上。噪声 p 由发动机附近的传感器来测量。电话扩音器收到部分寄生信号，例如 $kp(k<1)$，寄生信号加在有用信号 x 上，如图 6-1 所示。

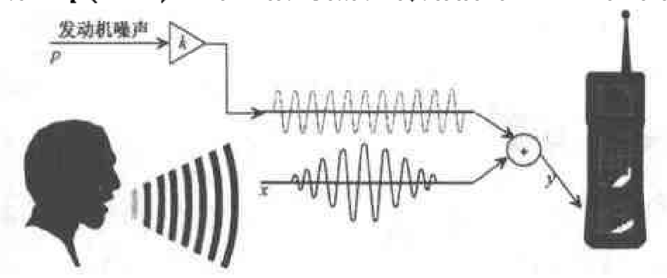


图 6-1　车载免提电话示意图

文件 teleph_ml.m 用一个线性神经元来减轻由于发动机噪声产生的干扰。将发动机噪声加入到神经元的输入，希望能预测噪声信号 y。神经元将取出部分与噪声 p 相关的信号或接近的信号。预测误差，也就是目标（噪声信号）和神经元输出信号之间的差，将组成有用信号的 x 值。

6.1　用 MATLAB 指令编制学习机

文件 teleph_ml.m 用“神经网络工具箱”中的 learnwh 函数来编制学习机。

hf_teleph.m

```
% interference cancellation
clear all, close all
% hands free telephone
t = 0:0.0005:1;
x = sin(sin(20*t).*t*200);

% plotting of useful signal x
figure(1)
plot(t, x)
axis([0 1 -1.2 1.2])
xlabel('time'), title('useful voice signal x'), grid

% parasitic signal
p = (square(20*pi*t)/2)+(0.2*sin(100*pi*t-100));
figure(2)
plot(t, p)
axis([0 1 -1.2 1.2])
xlabel('time'), title('the engine noise p'), grid

% noisy signal
```

```
figure(3)
z = x + 0.833*p;
plot(t, z)
xlabel('time'),title('noisy signal z = x + 0.833 p'), grid

% random initialization of the weights W and bias b
W = randn(1,1); b = randn(1,1);

weights_W = W; bias_b = b;

eta = 0.1; % adaptation gain

for i=1:length(t)
  % the neuron output
  y(i) = W * p(i) + b;

  % error output
  e(i) = z(i) - y(i);
  [dW,db] = learnwh(p(i),e(i),eta);

  % updating the weights W and bias b matrices
  W = W + dW; b = b + db;

  % saving the weights and bias matrices
  weights_W = [weights_W, W];
  bias_b = [bias_b, b];
end

% extracted signal
figure(4), plot(t, e), axis([0 1 -1.2 1.2])
xlabel('time'),title('extracted signal'), grid
% extraction error
figure(5), plot(t, x-e), xlabel('time'), title('extraction error'), grid
axis([0 1 -1 1])
```

图 6-2 至图 6-4 分别显示由电话复合扩音器捕获的有用信号 x、发动机噪声 p 以及噪声信号 $z=x+0.833p$。

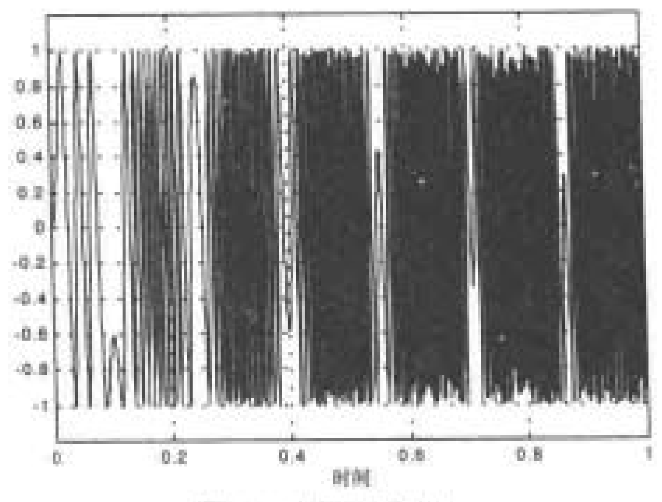


图 6-2　有用声音信号 x

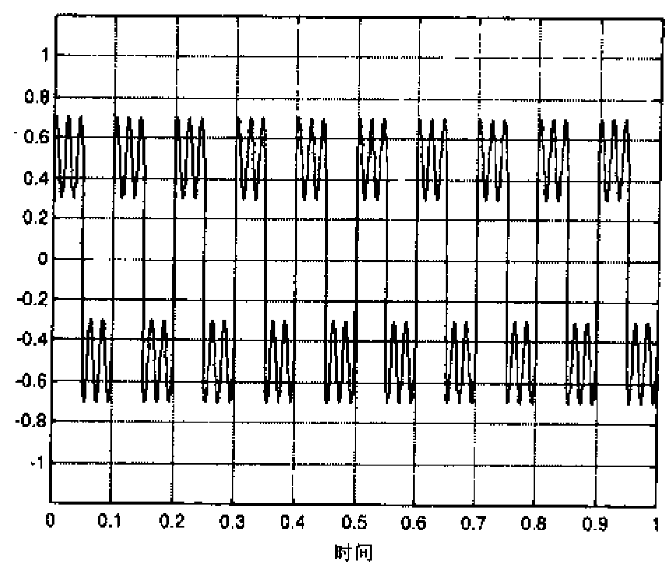


图 6-3　发动机噪声 p

图 6-4　噪声信号 $z=x+0.833p$

由自适应线性元件提取的信号非常接近有用信号 x（见图 6-5）。

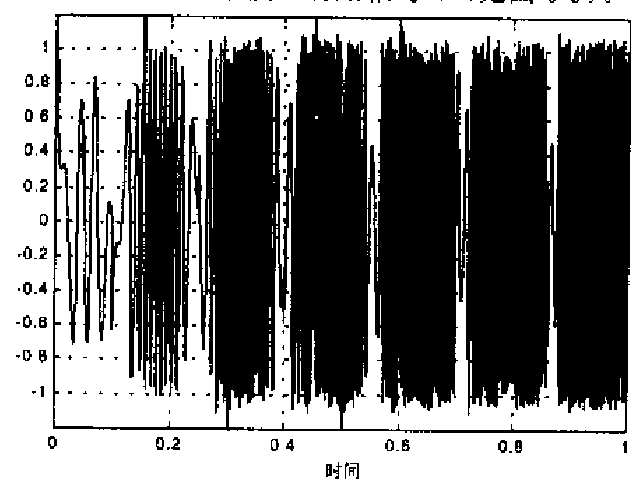


图 6-5　提取的信号

定义有用信号与提取信号之差的取出误差幅度在瞬间后大约为 0.1（见图 6-6）。

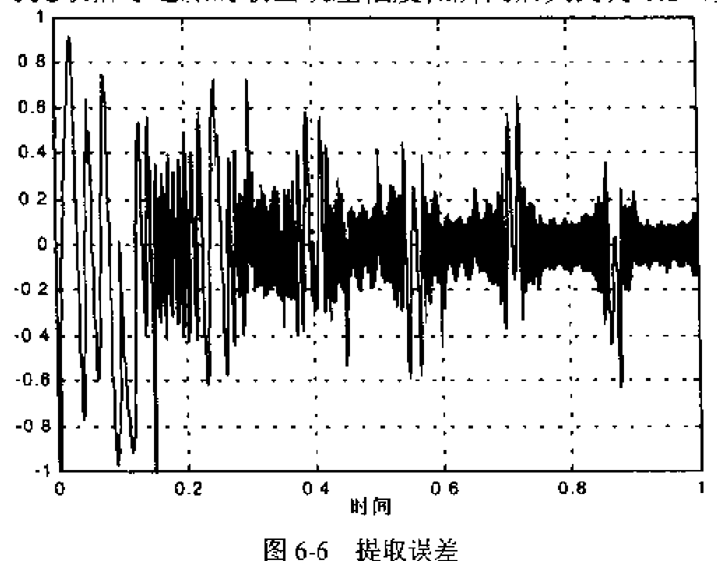


图 6-6　提取误差

最终的 weight 值和自适应线性元件偏差为：

```
w=
   0.9443
>>b
b=
-0.0566
```

6.2　在 SIMULINK 模型中使用 S 函数

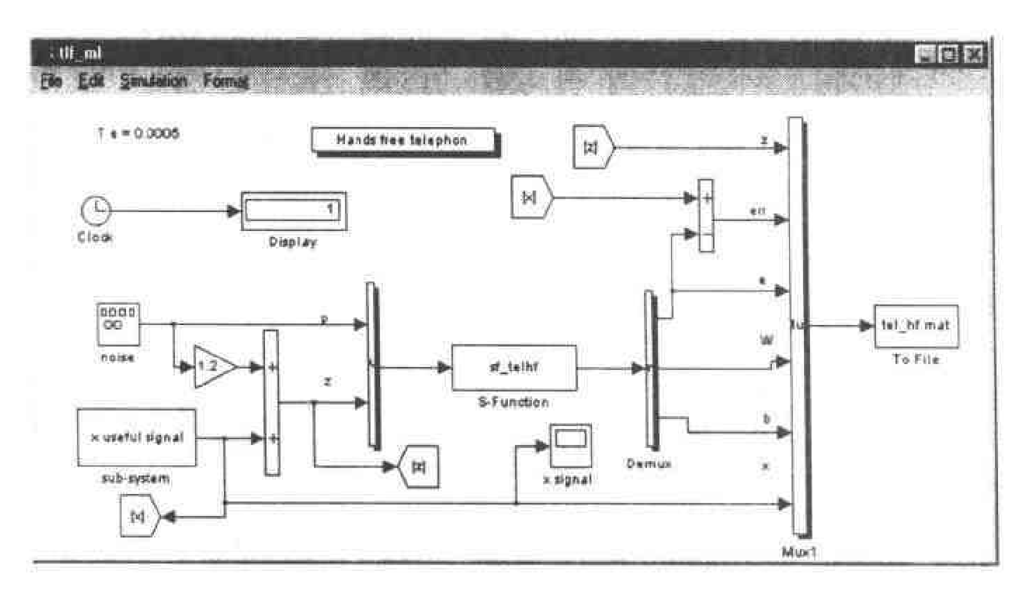


图 6-7　免提电话的原理图

有用声音信号由下面的子系统产生（见图 6-8）。

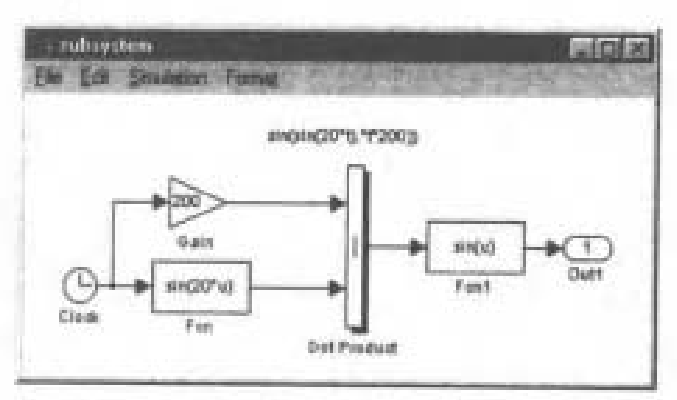


图 6-8 产生有用信号的子系统

S 函数的代码如下。使用"神经网络工具箱"的 learnwh 函数。作为 S 函数参数的采样周期为 T_s=0.000 5 s，这个变量受工作环境的影响。

S 函数 sf_telhf.m

```
% S-function : Adaline for the hands free telephone
function [sys,x0,str,ts] = sf_tel_ml(t,x,u,flag,Te)
global W b eta
switch flag,
case 0
  global W b eta
  [W,b] = rands(1,1);
  eta = 0.02;
  [sys,x0,str,ts] = Initialization(Te);
case 3
  p = u(1);
  z = u(2);
  y = W*p+b;
  e = z-y;
  [dw, db] = learnwh(p,e,eta);
  W = W + dw;
  b = b + db;
  sys = [e, W, b];
case {1,2,4,9}
  sys = [];
otherwise
  error(['Unhandled flag = ', num2str(flag)]);
end
function [sys,x0,str,ts] = Initialization(Te)
  sizes = simsizes;
  sizes.NumContStates = 0;
  sizes.NumDiscStates = 0;
  sizes.NumOutputs = 3;
  sizes.NumInputs = 2;
  sizes.DirFeedthrough = 0;
  sizes.NumSampleTimes = 1;
  sys = simsizes(sizes);
  x0 = [];
  str = [];
  ts = [Te 0];
```

文件 sgn_telhf.m 显示仿真结果。

sgn_telhf.m

```
% results display
% Adaline : S-fonction hands free telephone
load tel_hf.mat
z = signals(2,:);   % noisy signal
err = signals(3,:); % extraction error
e = signals(4,:);   % extracted signal
W = signals(5,:);   % Weight W
b = signals(6,:);   % bias b
x = signals(7,:);   % useful signal
figure(1), plot(z)
title('noisy signal')
axis([0 2000 -2.5 2.5])
figure(2), plot(err), axis([0 2000 -1 1])
xlabel('time'), title('extraction error')
figure(3)
plot(e)
axis([0 2000 -1.2 1.2])
xlabel('time')
title('extracted signal')
```

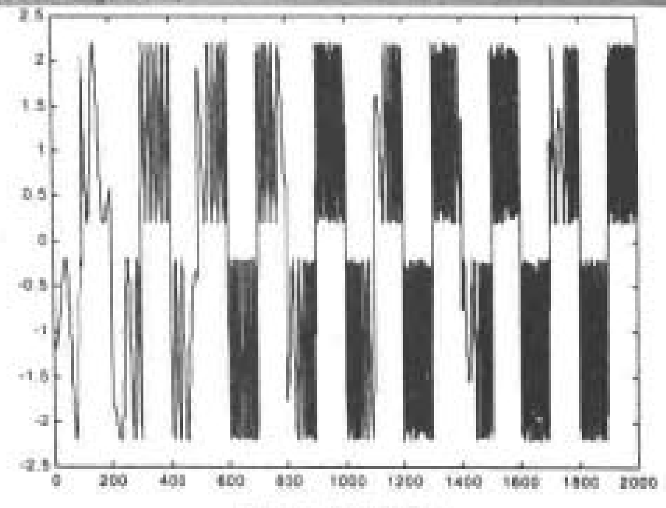


图 6-9　噪声信号

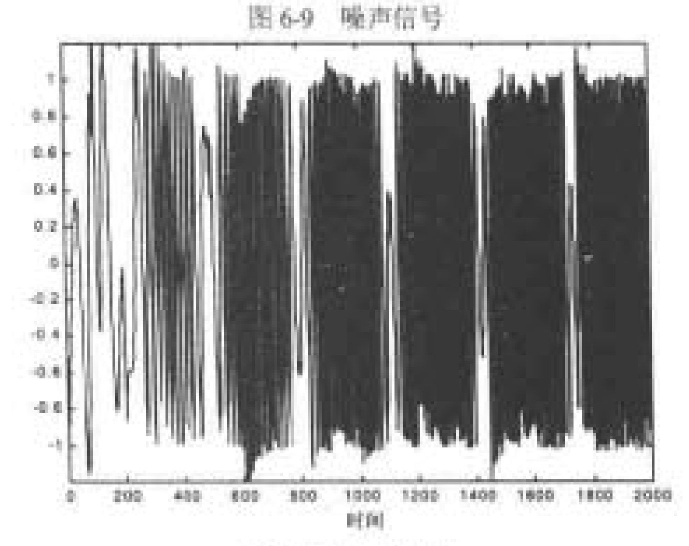
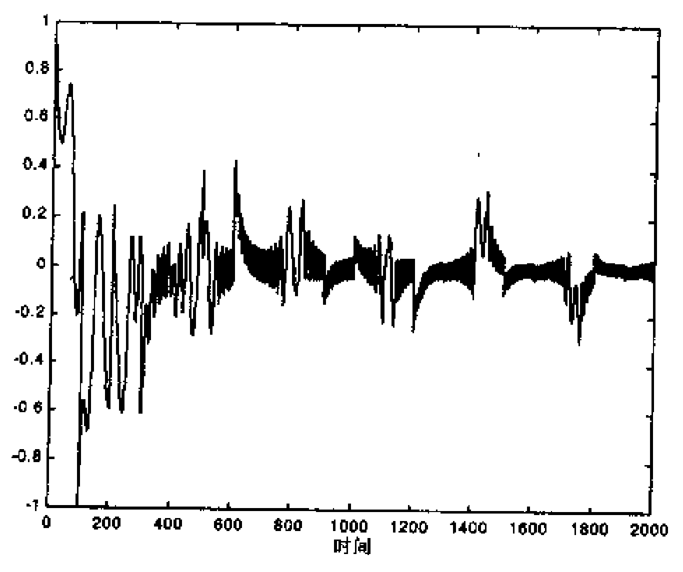


图 6-10　提取信号



1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
200
400
600
800
1000
1200
1400
1600
1800
2000
时间

图 6-11　提取误差

应用7　传输线上的回声抵消

本应用包括使用自适应滤波器来调节环境状态的特定特性。我们使用它来抵消传输线（比如电话线）上产生的回声（见图 7-1）。事实上，由于线长以及适应性的可能损失，一个发出的信号可能部分地传回至发射端，这样就产生令人不悦的回声现象。

传输线的特性对不同的连接方式是不同的，因此这也就是它的缺点。这里使用自适应滤波来限制这些缺点。

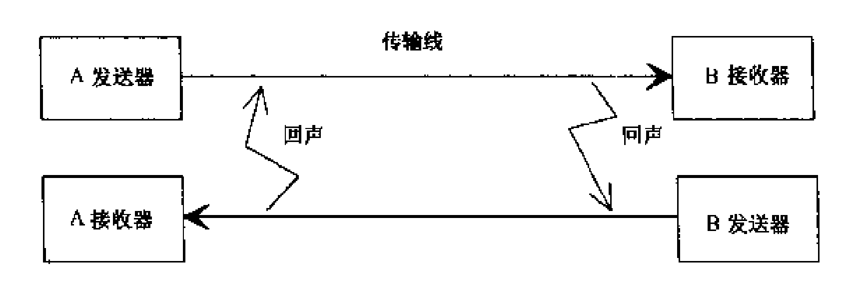


图 7-1　回声的产生

如果只考虑从 A 到 B 方向产生的回声抵消，可按图 7-2 所示加入自适应滤波器。

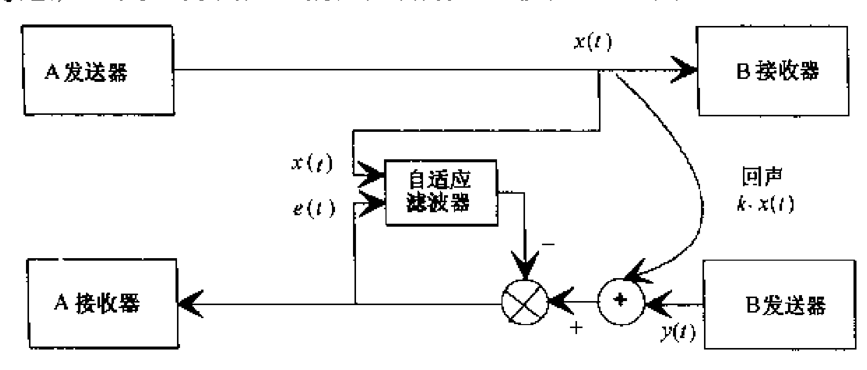


图 7-2　加入自适应滤波器的传输线

信号 $x(t)$和它的回声 $kx(t)$联系密切。另一方面，发送器信号 $y(t)$和回声信号 $kx(t)$非相关，于是从参考信号 $x(t)$中抵消 $e(t)$的滤波调节使 $e(t)$趋向于 $y(t)$，这样就衰减了反馈线上的回声信号。

7.1　传输线模型

对于传输线模型，只考虑由于信号传输持续时间而产生的传输延迟。回声信号由接收信号产生，它经过较小的失真后乘以一个衰减系数。整个 SIMULINK 模型如图 7-3 所示。

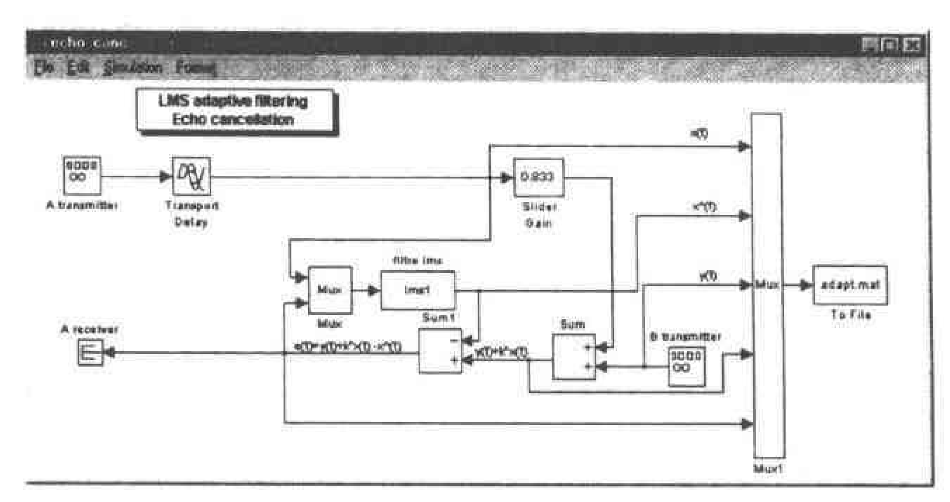


图 7-3 LMS 自适应滤波回声抵消的 SIMULINK 模型

在第一个例子中，回声信号发生衰减。控制在 10 ms 内的传输时间与长距离传输线上产生的延迟相比是很小的。只是因为“传输块延迟”的特定功能使输出信号只有在时间延迟后出现。

利用 S 函数得到自适应滤波器。这里传递采样周期为 T_s、自适应滤波系数为 δ、滤波器阶数为 n 等参数。

S 函数 lms1 执行使用梯度算法的自适应滤波器。

7.2 LMS 滤波，S 函数 lms1

除了系统参数，参数 T_s、delta 和 n 都将赋给 lms1 函数。初始阶段将输入矢量 x_c、系数矢量 h 以及误差标量 err 调为 0，调用系统初始函数。

滤波器计算 lms 在每个 T_s 采样按如下算法计算：

$$y(k) = h(k-1)x'(k)$$
$$h(k) = h(k-1) + \delta e(k)x(k)$$
$$e(k) = x_r(k) - y(k)$$

此应用中：

- $x(k) = u(1)$ S 函数的第一个输入；
- $x_r(k) = e(k) = y(k) + x(k) - \hat{x}(k) = u(2)$ 第二个输入；
- $y(k) = \hat{x}(k) = yf = sys$ S 函数输出。

lms1.m file

```
% LMS adaptive filtering
% Gradient algorithm

function [sys,x0,str,ts] = lms1(t,x,u,flag,Ts,delta,n)
global yf xe h err

switch flag
case 0            % initialization stage
   xe = zeros(1,n);
   h = zeros(1,n);
```

```
  err = 0;
  [sys,x0,str,ts] = Initialization(Ts);

case 3          % output calculation stage
  xe(1,n) = u(2);
  yf = 0;
  for j = 1:n
    yf = yf+h(1,j)'*xe(1,n-j+1);
    h(1,j) = h(1,j)+delta*err*xe(1,n-j +1);
  end
  err = u(1)-yf;
  for j = 1:n-1
    xe(1,j) = xe(1,j+1);
  end
  sys = yf;
case {1,2,4,9}  % non used stages
  sys = [];
otherwise
  error([num2str(flag)]);
end

% initialization function
  function [sys,x0,str,ts] = Initialization(Ts)
  sizes = simsizes;
  sizes.NumContStates = 0;
  sizes.NumDiscStates = 0;
  sizes.NumOutputs = 1;
  sizes.NumInputs = 2;
  sizes.DirFeedthrough = 0;
  sizes.NumSampleTimes = 1;
  sys = simsizes(sizes);
  x0 = [];
  str = [];
  ts = [Ts 0];
```

发送器 A 信号为具有单位幅度和 300 Hz 频率的正弦波信号，传输线导致了 1 ms 延迟，并且接收到的信号 $x(t)$产生幅度为 $0.833x(t)$的加在反馈信号上的回声。发送器 B 信号同样是具有单位幅度的正弦波信号，但频率为 1 000 Hz。为了进行仿真，传给 lms1 函数的参数取如下值：

$$\begin{cases} T_s = 0.000125\text{s} \\ \delta = 0.005 \\ n = 10 \end{cases}$$

由文件 echo1.m，得到持续时间为 0.5 s 的仿真结果。

echo1.m file

```
% Echo cancelation.
load adapt.mat
t=signals(1,:);
x=signals(2,:);
xe=signals(3,:);
```

```
y=signals(4,:);
yx=signals(5,:);
e=signals(6,:);
l=length(t);

% signals tracing
figure(1);
plot(t(1:floor(l/50)),x(1:floor(l/50))),hold on;
plot(t(1:floor(l/50)),y(1:floor(l/50))),hold off,grid;
axis([0 0.01 -1.2 1.2]);
gtext('x(t)'),gtext('y(t)');
title('x(t) and y(t) signals');
xlabel('Time'),ylabel('Magnitudes');

figure(2);
plot(t(1:floor(l/10)),xe(1:floor(l/10))),grid;
title('xf(t) filter output signal');
xlabel('Time'),ylabel('Magnitude');

figure(3);
plot(t(1:floor(l/20)),yx(1:floor(l/20)),'r'),grid;
title('kx(t)+ y(t)= transmitter signal + echo');
xlabel('Time'),ylabel('Magnitude');

figure(4);
plot(t(ceil(l-l/20):l),e(ceil(l-l/20):l)),grid;
title('e(t) signal');
xlabel('Time'),ylabel('Magnitude');
clear t;
```

◆ A、B 发送器信号（见图 7-4）

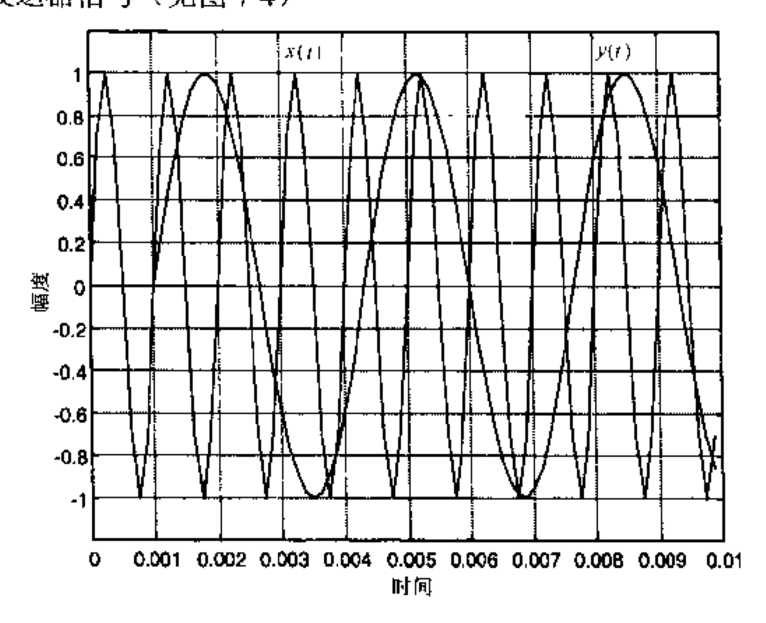


图 7-4　$x(t)$和 $y(t)$信号

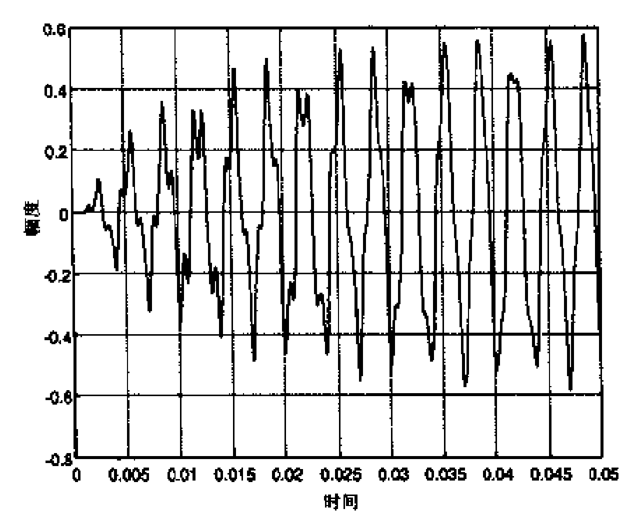


图 7-5 $x_f(t)$滤波器输出信号

由于小的自适应系数λ，输出滤波慢慢地调整至信号 $kx(t)$。在达到大约 63 %稳态幅度（即幅度约为 0.5 V）的输出信号时测量自适应时间常数，得到$\tau \approx 20$ ms，也就是说时间常数为 $160T_s$，T_s=0.125 ms。

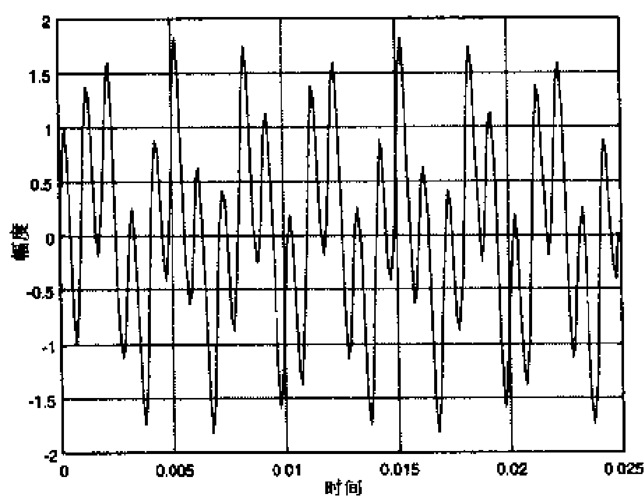


图 7-6 $y(t)+kx(t)$=发送器信号+回声

发送器 B 信号被回声严重干扰。在反馈线上，得到一个只与回声信号弱耦合的滤波信号。

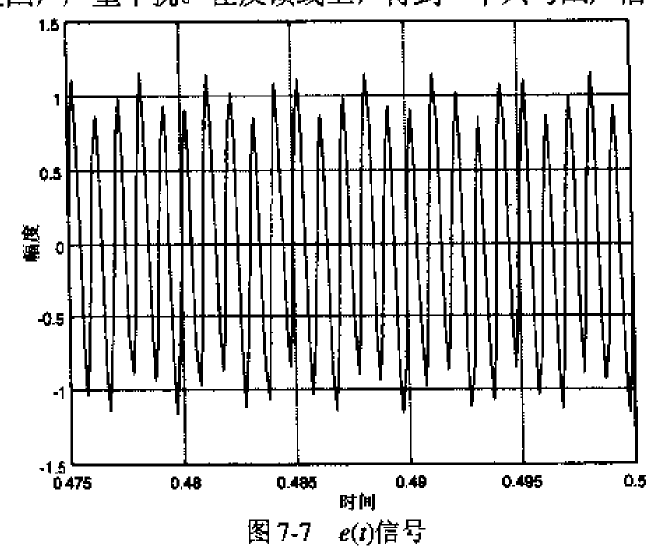


图 7-7 $e(t)$信号

如果技术限制了材料的使用，改善因素将简单地由增加的滤波器阶数组成。

◆ n=100 时的仿真（见图 7-8）

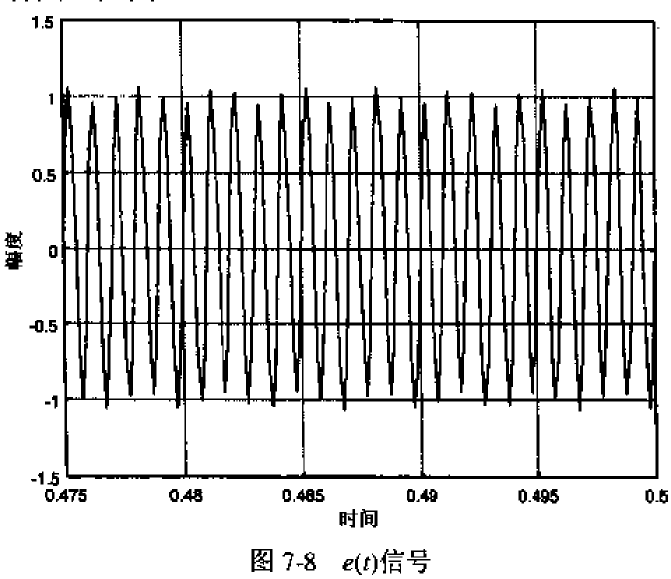


图 7-8 $e(t)$信号

7.3 RLS 滤波，S 函数 rls1

执行自适应滤波的 S 函数可根据 RLS 滤波技术得到。除了系统参数，参数 T_s和 n 也被赋给 rls1 函数。初始阶段将输入矢量 x_e、增益矢量 g、系数矢量 h 以及误差标量 err 调为 0。变量逆矩阵初始化为 10000I，I 为（n,n）维单位矩阵。接着调用系统初始化函数。rls 滤波计算根据在每个 T_s 采样周期按如下算法计算：

$$g(k)=\frac{C_{xx}^{-1}(k-1)x^{\mathrm{T}}(k)}{1+x(k)C_{xx}^{-1}(k-1)x^{\mathrm{T}}(k)}$$

$$C_{xx}^{-1}(k)=C_{xx}^{-1}(k-1)-g^{\mathrm{T}}(k)x(k)C_{xx}^{-1}(k-1)$$

$$e(k)=y(k)-x(k)h^{\mathrm{T}}(k-1)$$

$$h(k)=h(k-1)+g(k)e(k)$$

$$y(k)=h(k)x^{\mathrm{T}}(k)$$

此应用中：

- $x(k)=u(1)$ S 函数的第一个输入；
- $x_r(k)=e(k)=y(k)+kx(k)-\hat{x}(k)=u(2)$ 第二个输入；
- $y(k)=\hat{x}(k)=yf=\mathrm{sys}$ S 函数输出。

rls1.m file

```
% RLS adaptive filtering
% MCR Algorithm

function [sys,x0,str,ts] = rls1(t,x,u,flag,Ts,n)

% Global variables
global yf xe h err i Cxx g
```

```
switch flag

% initialization stage
case 0
  i = n;
  Cxx = 10000*eye(n);
  g = zeros(1,n);
  xe = zeros(1,n);
  h = zeros(1,n);
  err = 0;
  [sys,x0,str,ts] = Initialization(Ts);
% output stage calculation
case 3
  xe(1,n) = u(2);
  g = (Cxx*xe'/(1+xe*Cxx*xe'))';     % Adaptative gain
  err = u(1)-h*xe';                  % A priori error
  h = h+err*g;                       % Coefficients
  Cxx = Cxx-g'*xe*Cxx;               % Correlation matrix
  yf = h*xe';                        % Filter output
  i = i+1;

  for j = 1:n-1
    xe(1,j) = xe(1,j+1);
  end
  sys = yf;

% non used stages
case {1,2,4,9}
  sys = [];

otherwise
  error(['Unhandled flag = ',num2str(flag)]);
end

% initialization function
  function [sys,x0,str,ts] = Initialization(Ts)
  sizes = simsizes;
  sizes.NumContStates = 0;
  sizes.NumDiscStates = 0;
  sizes.NumOutputs = 1;
  sizes.NumInputs = 2;
  sizes.DirFeedthrough = 0;
  sizes.NumSampleTimes = 1;
  sys = simsizes(sizes);
  x0 = [];
  str = [];
  ts = [Ts 0];
```

为进行仿真，赋给 rls1 函数的参数取如下值：

$$\begin{cases} T_s = 0.000125\,\text{s} \\ n = 10 \end{cases}$$

对于同样的驱动信号，由文件 echo1.m 可得到持续时间为 0.5 s 的仿真结果。

可以看到，滤波器输出信号的收敛比 LMS 滤波器快，这要由变量矩阵的初始值来决定。另一方面，因为自适应增益衰弱得很快，这个滤波器更难追踪非稳定信号的变量，所以 LMS 滤波器更适合这种类型的应用。

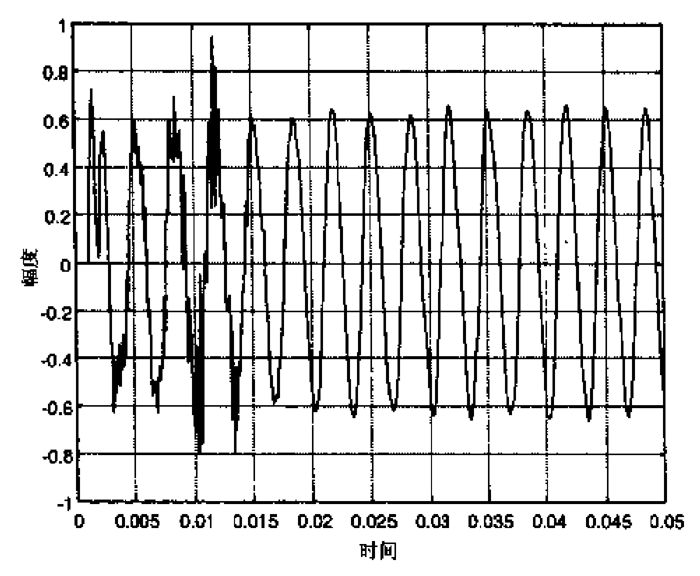


图 7-9 滤波输出信号 $x_f(t)$

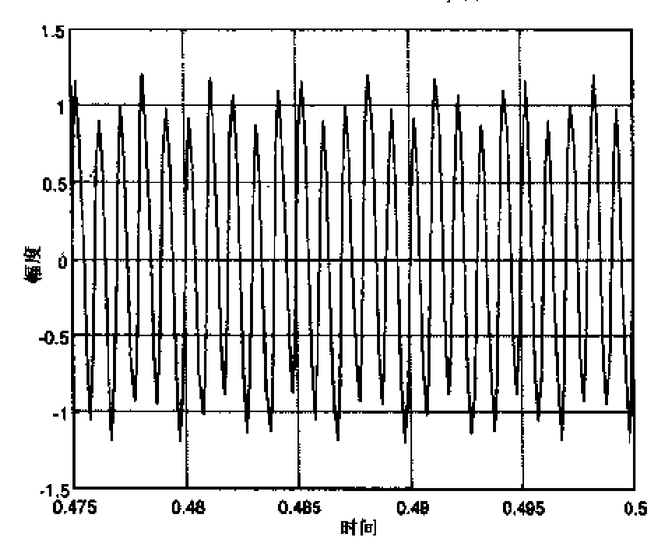


图 7-10 $e(t)$信号

应用8　导管内的噪声抵消

本应用是由使用自适应滤波器抵消通风管道内风扇产生的噪声构成的。风扇产生的声波很快在导管内传播，这放大了一些噪声且引入了共鸣（见图8-1）。

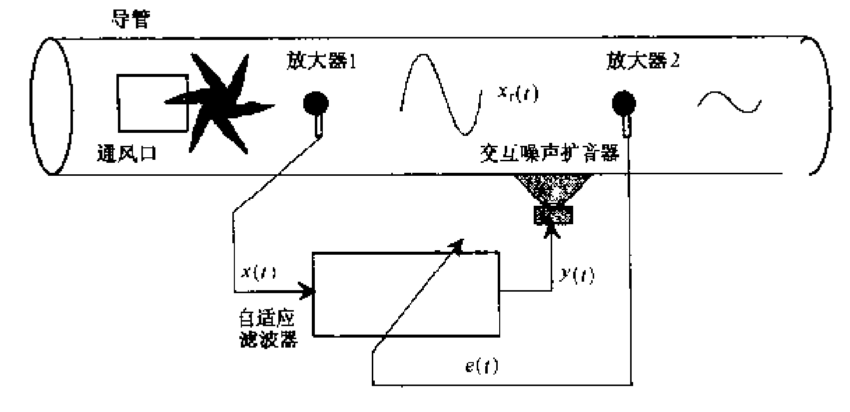


图8-1　导管内部结构示意图

导管噪声被扩音器1捕获以便作为自适应滤波器输入信号 $x(t)$。滤波器输出信号 $y(t)$产生抵消导管内残余噪声的反噪声。扩音器2用来捕获残余噪声，此噪声用于调节滤波系数，使 $e(t)$趋于0。

8.1　导管模型

风扇产生的噪声根据旋转速度和叶片数目而产生一个基本频率。假定6个叶片的风扇以25 rps的速度旋转，将相应产生150 Hz的噪声基本频率。

声音谐波由基本信号通过一个“look-up counts”非线性函数产生。增益项表示扩音器1位置和扩音器反作用间的衰减。函数“transport delay”表示声音信号的传输时间。整个SIMULINK模型如图8-2所示。

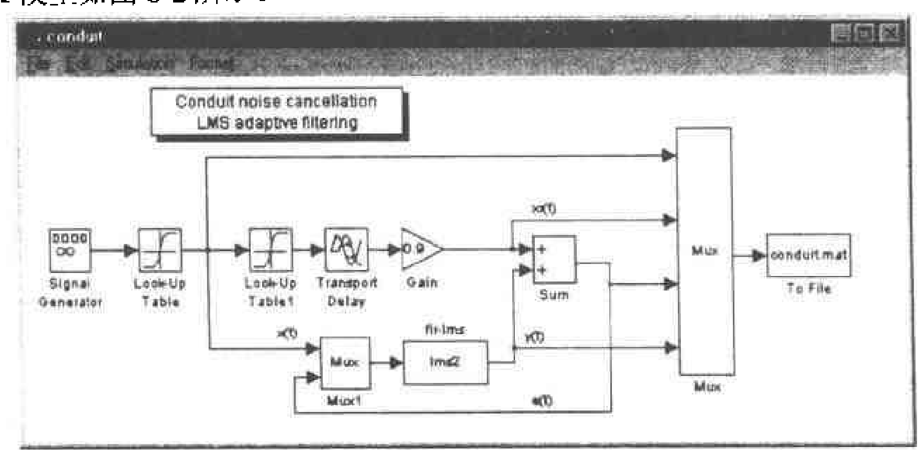


图8-2　导管噪声抵消LMS自适应滤波的SIMULINK模型

自适应滤波器由 S 函数实现，这里将把采样周期 T_s、自适应滤波系数 δ 和滤波器阶数 n 等参数赋给 S 函数。自适应滤波器使用梯度算法，相应的文件为 lms2.m。

8.2 LMS 滤波，S 函数 lms2

除了系统参数，参数 T_s、*delta*、n 都赋给函数 lms2。初始化阶段使输入矢量 *xe*、系数矢量 h 和误差标量 err 为 0，然后调用系统初始化函数。lms 滤波计算根据 T_s 采样周期按如下算法计算：

$$y(k) = h(k-1)x'(k)$$
$$h(k) = h(k-1) + \delta e(k)x(k)$$

在本应用中：

- $x(k) = u(1)$ S 函数的第 1 个输入；
- $e(k) = u(2)$ 第 2 个输入；
- $y(k) = sys = yf$ S 函数输出。

lms2.m file

```
% LMS Adaptive filtering
% Gradient algorithm
function [sys,x0,str,ts] = lms1(t,x,u,flag,Ts,delta,n)
global err h xe yf

switch flag,
case 0              % Initialization stage
  xe = zeros(1,n);
  h = zeros(1,n);
  err = 0;
  [sys,x0,str,ts] = Initialization(Ts);
case 3              % Output calculation stage
xe(1,n) = u(1);
  yf = h*xe';
  err = -u(2);
  h = h+delta*err*xe;
  for j = 1:n-1,
     xe(1,j) = xe(1,j+1);
  end
sys = yf;
case {1,2,4,9}  % Non used stages
  sys = [];
otherwise
  error(['Unhandled flag = ',num2str(flag)]);
end

% Initialization function
  function [sys,x0,str,ts] = Initialization(Ts)
  sizes = simsizes;
  sizes.NumContStates = 0;
  sizes.NumDiscStates = 0;
  sizes.NumOutputs = 1;
```

```
sizes.NumInputs = 2;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [Ts 0];
```

为了进行仿真，赋给 S 函数 lms2 的参数为：

$$\begin{cases} T_s = 0.1\,\text{ms} \\ \delta = 0.000\,2 \\ n = 100 \end{cases}$$

考虑到谐波数，选择的采样周期与基本采样周期相比较小。选择的调节系数 δ 对减小残余误差作用不大。另一方面，自适应滤波时间常数较大，由于噪声稳定，所以没有影响。

因为梯度算法与其他自适应的算法相比几乎无需计算，因此我们可以擅自选择高阶，使用的算法需要 $2n$ 次乘加运算。在数字信号处理器中，一个乘加运算需要 1 MPU 指令周期。此处，快速计算所需的时间为 $\frac{T_s}{2n} = 0.5\,\mu\text{s}$ 的 MPU 周期，也就是通常的频率至少为 2 MHz 的处理器。

由 conduit1.m 文件，在 0.1 s 持续时间下仿真结果被执行。

conduit_1.m file

```
% conduit noise cancellation

load conduit.mat
t = Signals(1,:);
x = Signals(2,:);
xr = Signals(3,:);
e = Signals(4,:);
y = Signals(5,:);

% Signals representation
figure(1)
plot(t,x), grid
xlabel('Time'), ylabel('Amplitude in V')
title('Noise measured by the microphone 1')
figure(2)
plot(t,y), grid
xlabel('Time'), ylabel('Amplitude in V')
title(' y(t) counternoise ')
figure(3)
plot(t,e), grid
xlabel('Time'), ylabel('Amplitude in V')
title('Residual error measured by the microphone 2')

% Normalized spectra representation
figure(4)
N = 512;
xf = abs(fft(x,N))*2/N;
```

```
f = (0:N-1)/(N*Ts);
stem(f(1:50),xf(1:50))
title('x(t) spectrum')
xlabel('Frequency in Hz')
ylabel('Components amplitude')
figure(5)
l = length(e);
ef = abs(fft(e(l-N:N),N))*2/N;
stem(f(1:50),ef(1:50))
title('e(t) spectrum')
xlabel('Frequency in Hz')
ylabel('Components amplitude')

% Power spectral densities
figure(6)
psd(x,N,1/Ts)
title('x(t) power spectral density')
xlabel('Frequency in Hz')
ylabel('Modulus in dB')
figure(7)
psd(e(l-N:N),N,1/Ts)
title('e(t) power spectral density')
xlabel('Frequency in Hz')
ylabel('Modulus in dB')

% Noises powers
Px = sum(xf.^2)/4;
Pe = sum(ef.^2)/4;
disp([' Noise power before filtering = 'num2str(Px)]);
disp([' Noise power after filtering = 'num2str(Pe)]);
disp([' Improvement factor 'num2str(Px/Pe)]);
```

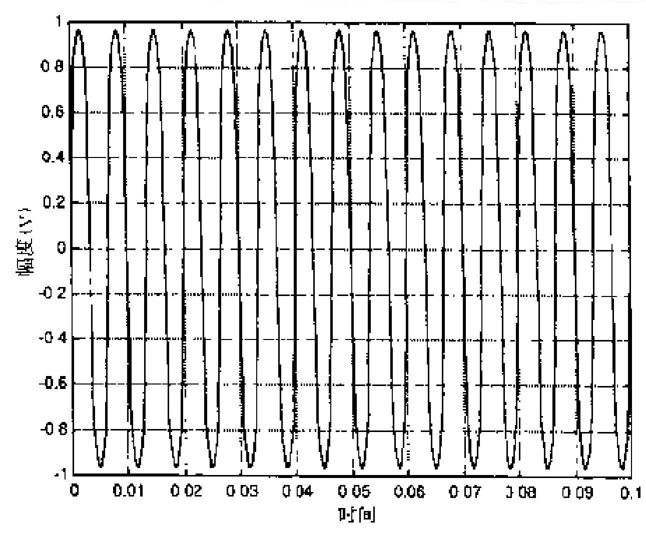


图 8-3 由扩音器 1 测量的噪声

滤波器产生的交互噪声与扩音器 1 测量的噪声反相（见图 8-4）。

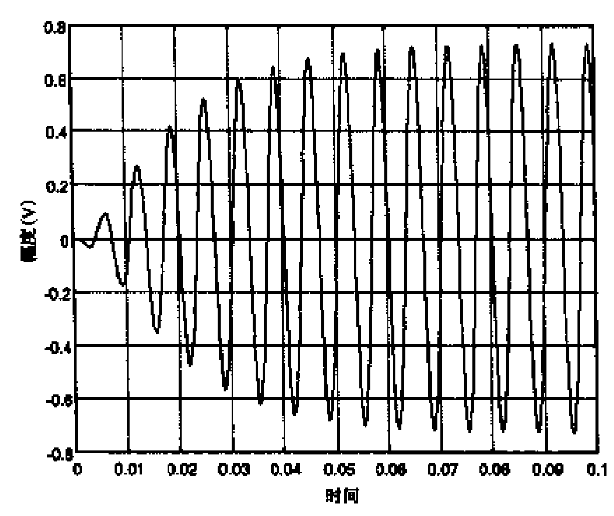


图 8-4　交互噪声 $y(t)$

建立的滤波器时间相对较长，约为 25 ms，也就是 $250T_s$（见图 8-5），这是由于自适应系数 δ 较小。

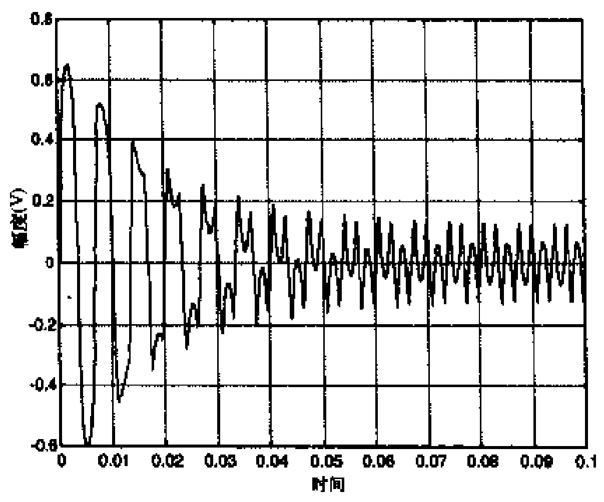


图 8-5　扩音器 2 测量的残余误差

使用函数 fft 可得到测量的噪声 $x(t)$和残余噪声频谱 $e(t)$。$x(t)$频谱的基波出现在 150 Hz 处，其幅度为 0.9，谐波在非线性部分出现，特别是谐波 3（见图 8-6）。

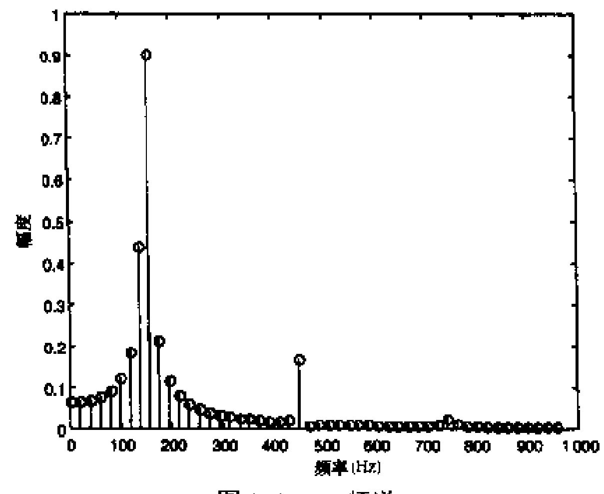


图 8-6　$x(t)$频谱

$e(t)$频谱幅度很大，接近白噪声（见图 8-7）。

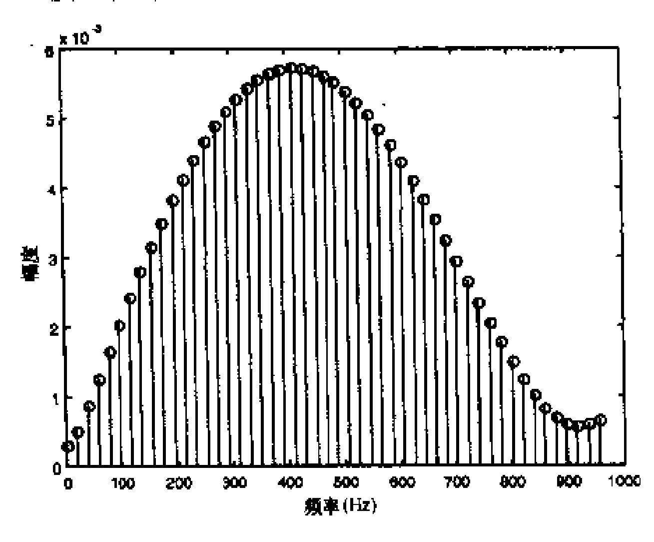


图 8-7 $e(t)$频谱

功率谱密度（见图 8-8），或者说功率在频率上的分布，是由函数 psd 得到的。

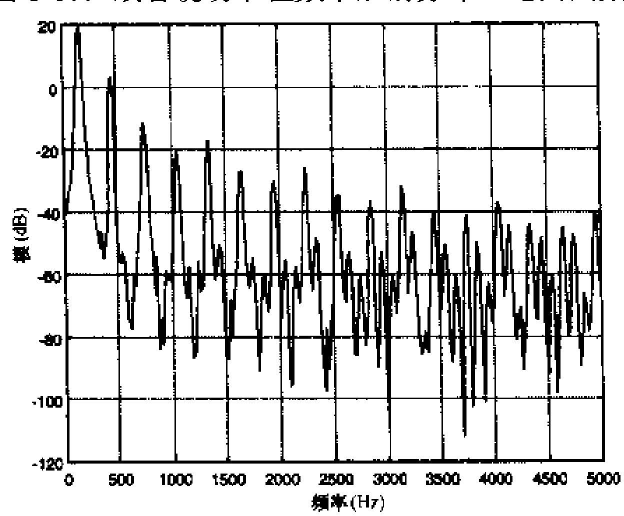


图 8-8 $x(t)$功率谱密度

$e(t)$残余噪声功率的主要部分在 0~1000 Hz 的频带区（见图 8-9）。

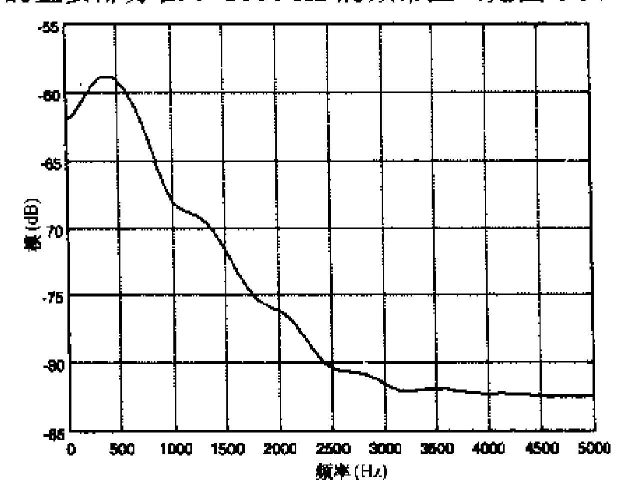


图 8-9 $e(t)$功率谱密度

整个信号功率由频谱上的每个分支功率相加而得到。

```
Noise power before filtering=0.59857
Noise power after filtering=0.00038414
Improbement factor 1558.2102
```

通过将自适应增益划分为 10 而得到新的仿真。赋给 S 函数的新参数为 T_s=0.1ms，δ=0.002，n=100。

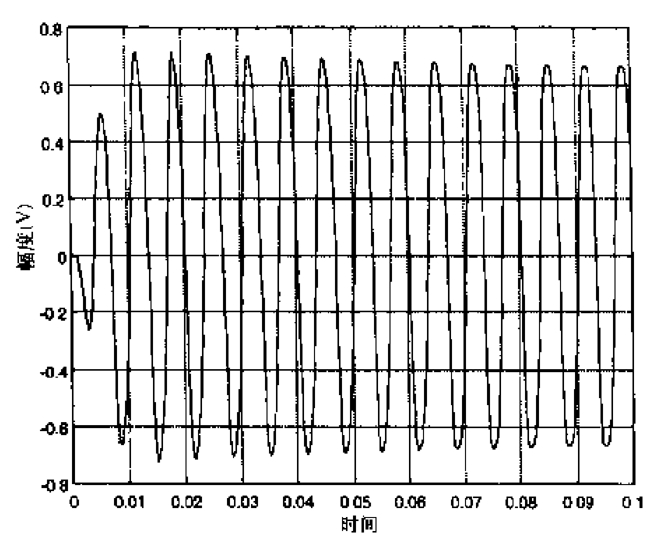


图 8-10　交互噪声 $y(t)$

我们看到滤波响应时间明显减小。

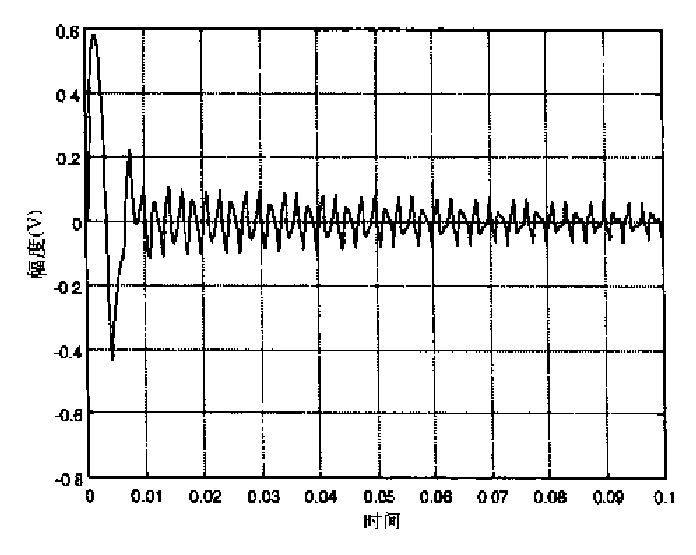


图 8-11　由扩音器 2 测量的残余误差

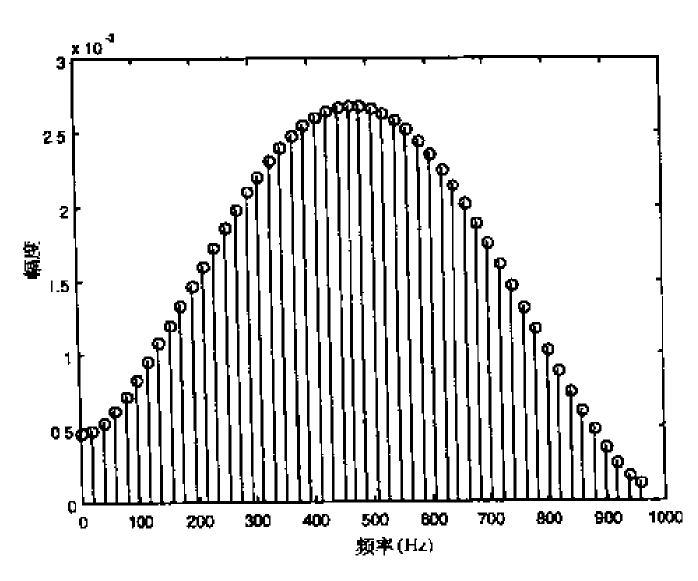


图 8-12　$e(t)$频谱

残余噪声频谱的整个形状不改变，但谱线的幅度缩小将近一半。

```
Noise power before filtering = 0.59857
Noise power after filtering = 9.0992e-005
Improvement factor 6578.2929
```

残余的噪声功率被接近 4 的系数划分。如果继续减小自适应增益，可以看到自一特定点，残余噪声功率又开始增加。实际上，自适应增益的选择由检验序列的折衷产生。

8.3　RLS 滤波，S 函数 rls2

仿真自适应滤波器的 S 函数根据最小二乘平方技术得到。除了系统参数，参数 T_s和 n 都要赋给函数 rls2。初始化阶段使输入矢量 xe、增益矢量 g、系数矢量 h 和误差标量 err 为 0。变量矩阵初始化为 10000I，I 为（n,n）单位矩阵。然后调用系统初始化函数。

RLS 滤波器根据如下算法在每采样周期 T_s时计算：

$$g(k)=\frac{C_{xx}^{-1}(k-1)x^{\mathrm{T}}(k)}{1+x(k)C_{xx}^{-1}(k-1)x^{\mathrm{T}}(k)}$$

$$C_{xx}^{-1}(k)=C_{xx}^{-1}(k-1)-g^{\mathrm{T}}(k)x(k)C_{xx}^{-1}(k-1)$$

$e(k)$由扩音器 2 测得：

$$h(k)=h(k-1)+g(k)e(k)$$

$$y(k)=h(k)x^{\mathrm{T}}(k)$$

在本应用中：

- $x(k)=u(1)$　S 函数的第一个输入；
- $x_r(k)=e(k)=u(2)$　第二个输入；
- $y(k)=sys=yf$　S 函数输出。

rls2.m file

```
% RLS adaptive filtering
% MCR Algorithm
```

```
function [sys,x0,str,ts] = rlsl(t,x,u,flag,Ts,n)
global  yf xe h err i Cxx g

switch flag,
case 0           % Initialization stage
  i = n;
  Cxx = 10000*eye(n);
  g = zeros(1,n);
  xe = zeros(1,n);
  h = zeros(1,n);
  err = 0;
  [sys,x0,str,ts] = Initialization(Ts);
case 3           % Output calculation stage
  xe(1,n) = u(1);
  g = (Cxx*xe'/(1+xe*Cxx*xe'))'; % Adaptation gain
  err = -u(2)      ;             % A priori error
  h = h+err*g;                   % Coefficients
  Cxx = (Cxx-g'*xe*Cxx);          % Correlation matrix
  yf = h*xe';                    % Output filter
  i = i+1;
  for j = 1:n-1
    xe(1,j) = xe(1,j+1);
  end;
  sys = yf;

case {1,2,4,9}  % Non used stages
  sys = [];
otherwise
  error(['Unhandled flag = ',num2str(flag)]);
end
% Initialization function
  function [sys,x0,str,ts] = Initialization(Ts)
  sizes = simsizes;
  sizes.NumContStates = 0;
  sizes.NumDiscStates = 0;
  sizes.NumOutputs = 1;
  sizes.NumInputs = 2;
  sizes.DirFeedthrough = 0;
  sizes.NumSampleTimes = 1;
  sys = simsizes(sizes);
  x0 = [];
  str = [];
  ts = [Ts 0];
```

◆ SIMULINK 模型（见图 8-13）

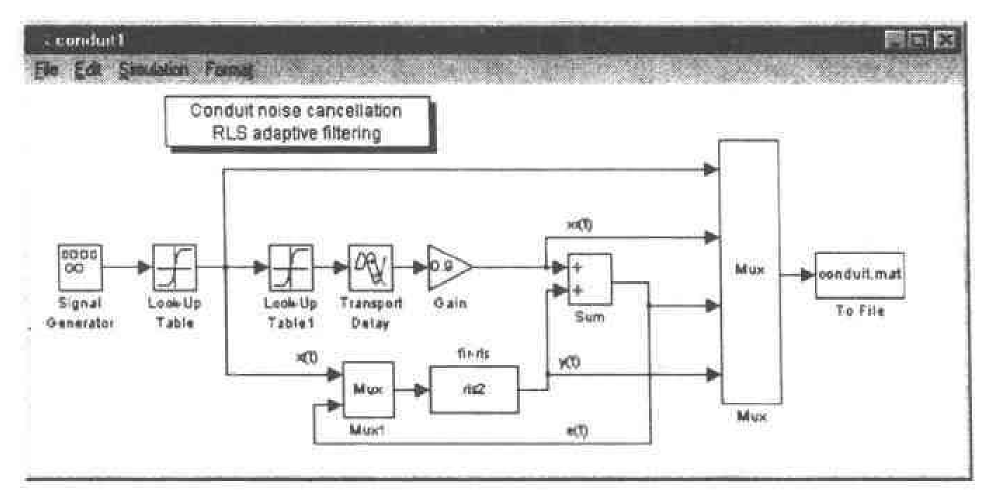


图 8-13　导管噪声抵消 RLS 自适应滤波的 SIMULINK 模型

为进行仿真，赋给 S 函数 rls2 的参数为

$$\begin{cases} T_s = 0.1\text{ms} \\ n = 10 \end{cases}$$

在 0.06 s 持续时间下执行的仿真结果由文件 conduit_2.m 得到。

conduit_2.m file

```
% conduit noise cancellation

load conduit.mat
t = Signals(1,:);
x = Signals(2,:);
xr = Signals(3,:);
e = Signals(4,:);
y = Signals(5,:);
% Signals representation
figure(2)
plot(t,y),hold on, plot(t,xr,'r:'), hold off
xlabel('Time'), ylabel('Amplitude in V')
title(' Noise and counternoise')
gtext('\leftarrowNoise'), gtext('\leftarrowCounternoise')
figure(3), plot(t,e), grid
xlabel('Time'), ylabel('Amplitude in V')
title('Residual error measured by the microphone 2')
% Power spectral densities
figure(6)
N = 512; l = length(e); psd(x,N,1/Ts)
title('Power spectral density of x(t)')
xlabel('Frequency in Hz'), ylabel('Modulus in dB')
figure(7), psd(e(l-N:N),N,1/Ts)
title('Power spectral density of e(t)')
xlabel('Frequency in Hz'), ylabel('Modulus in dB')
% Powers
xf = abs(fft(x,N))*2/N;
ef = abs(fft(e(l-N:N),N))*2/N;
Px = sum(xf.^2)/4; Pe = sum(ef.^2)/4;
disp([' Noise power before filtering = 'num2str(Px)]);
```

```
disp([' Noise power after filtering = 'num2str(Pe)]);
disp([' Improvement factor 'num2str(Px/Pe)]);
```

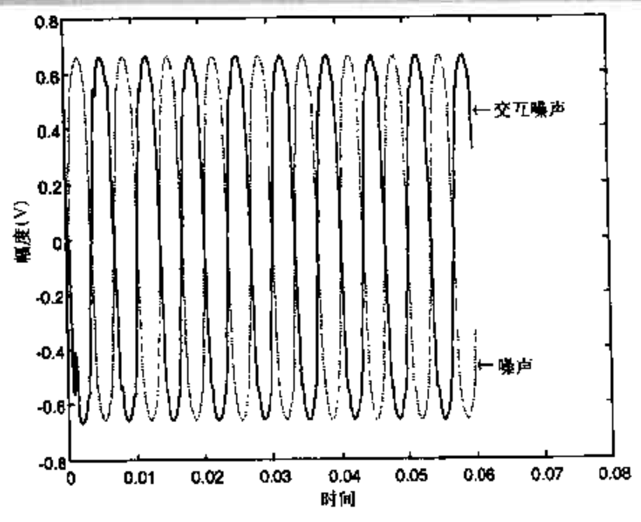


图 8-14　噪声和交互噪声

残余误差在 10 ms 内收敛于 0，这证明了 RLS 滤波器在 LMS 滤波器上的收敛速度。

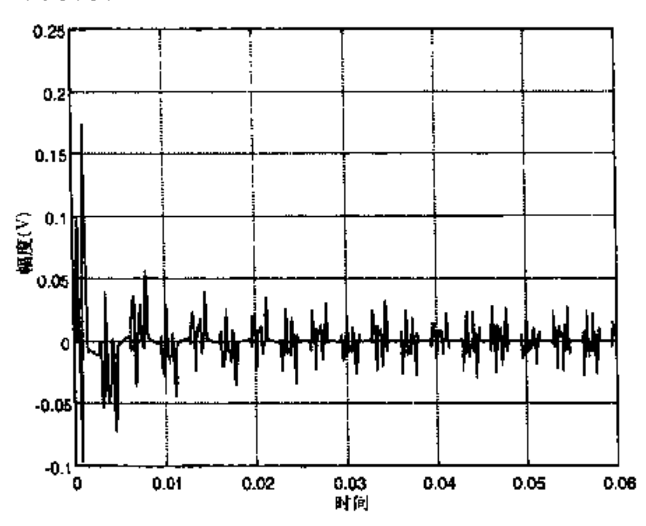


图 8-15　扩音器 2 测量的残余误差

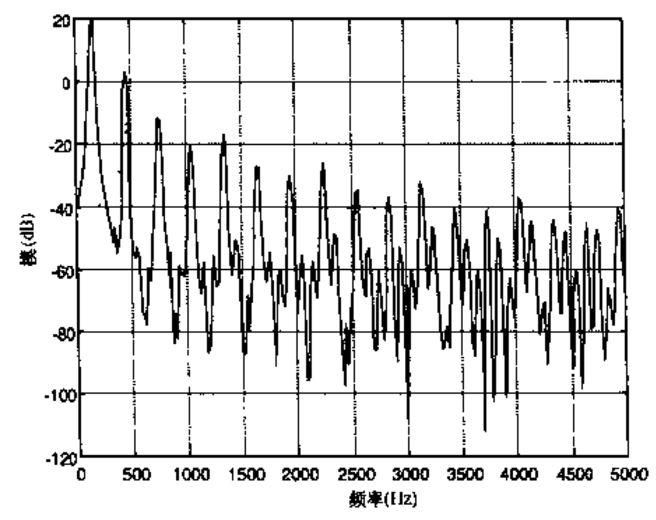


图 8-16　$x(t)$功率谱密度

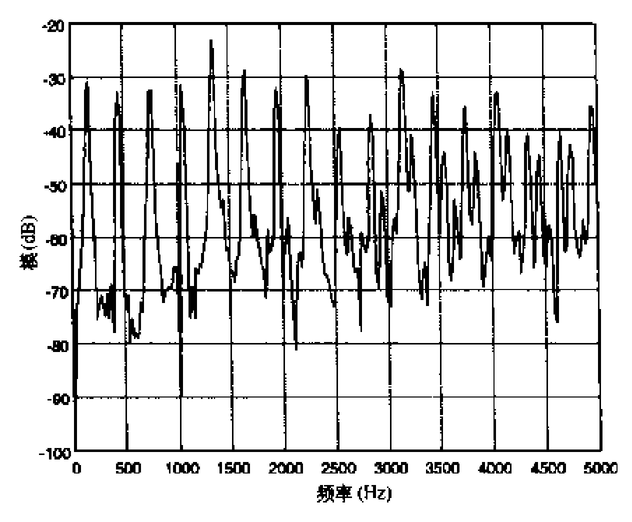


图 8-17　$e(t)$功率谱密度

残余噪声功率谱密度呈现了与测量噪声功率谱密度相比在大约-50 dB 下的突变。

```
Noise power before filtering=0.59857
Noise power after filtering=0.0001005
Improvement factor 5955.7654
```

8.4　复合噪声滤波

通过压缩由 3 个信号和形成的复合噪声 $x(t)$来进行最后的仿真：

- 幅度为 2V、频率为 150 Hz 的正弦信号；
- 幅度为 1V、频率为 300 Hz 的正弦信号；
- 幅度为 1V、频率为 1 000 Hz 的方波信号。

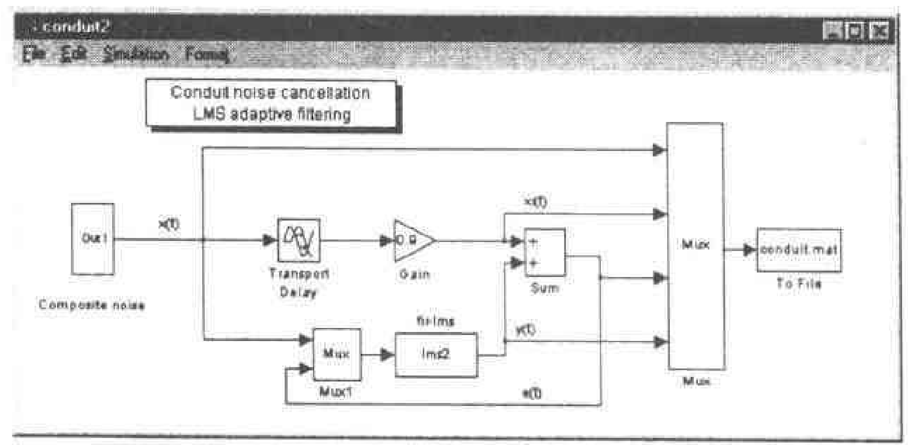


图 8-18　抵消导管噪声的 LMS 自适应滤波

文件 conduit_2.m 用来显示仿真结果。为了进行仿真，赋给 S 函数 lms2 的参数为

$$\begin{cases} T_s = 0.1\,\text{ms} \\ \delta = 0.0001 \\ n = 100 \end{cases}$$

因为滤波器不稳定，所以自适应增益取值相对较小。

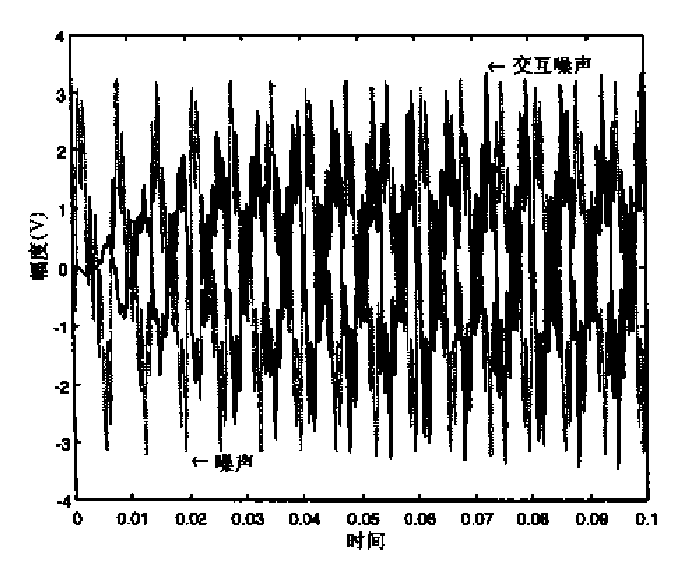


图 8-19　噪声和交互噪声

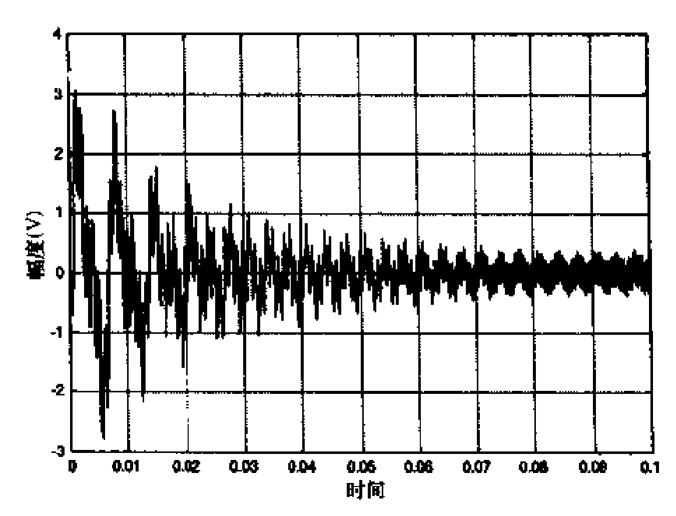


图 8-20　扩音器 2 测量的残余误差

交互噪声部分被抑制。

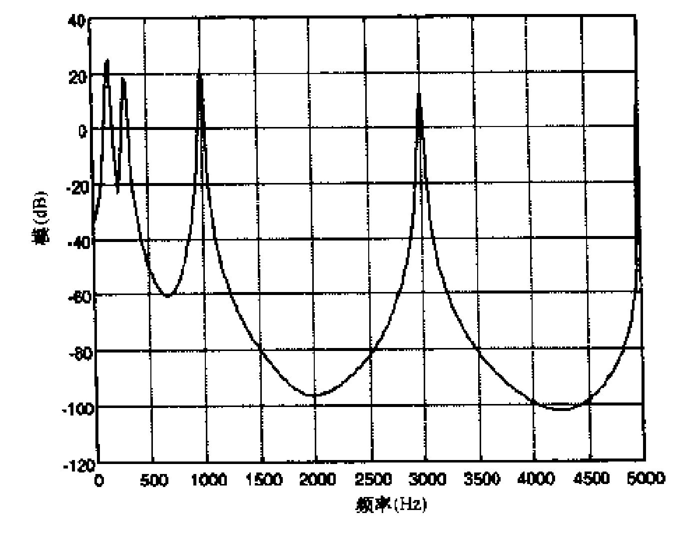


图 8-21　$x(t)$功率谱密度

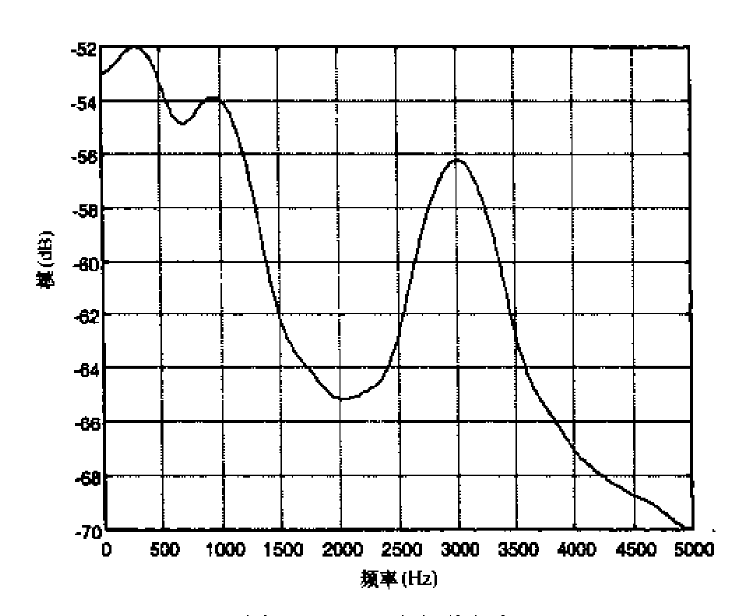


图 8-22　$e(t)$功率谱密度

由于正弦信号，噪声 $x(t)$的功率谱密度在频率 150 Hz 和 300 Hz 处都产生谱线；由于矩形脉冲和奇次谐波，在频率 1 000 Hz、3 000 Hz 和 5 000 Hz 处产生谱线。残余噪声功率谱密度产生同样的谱线，但衰减约 70 dB。

```
Noise power before filtering = 3.4473
Noise power after filtering = 0.0050058
Improvement factor 688.656
```

应用 9　对称二进制信道的均衡

本应用由得到均衡的信道构成。用于二进制信号传输的该信道引起发散作用，这加宽了二进制脉冲并产生符号间干扰现象（见图 9-1）。

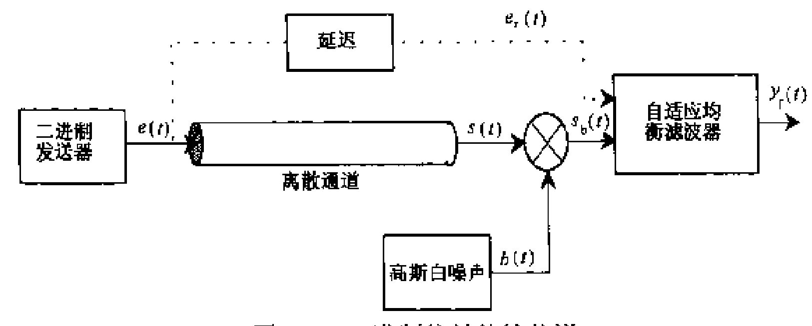


图 9-1　二进制信号传输信道

每当在非理想或特性未知（特别是高速率连接时）的信道上进行传输时，自适应均衡器即体现其重要作用。本应用中，均衡器计算时间是一个重要参数，我们考虑的是 LMS 型滤波器。

9.1　随机二进制序列的产生

为模拟二进制发送器，需产生一个伪随机二进制序列。因此，采用阶数为 10 的递进多项式，也就是 1024 比特的序列：

$$P(z)=1+z^{-7}+z^{-10}$$

描述 gene_sbpa.m 函数，按如下次序传递 4 个参数给函数 gene_sbpa.m：

- A　以 V 为单位的二进制水平幅度；
- T_s　二进制序列的采样周期；
- b　附加的高斯白噪声的均方差；
- l　二进制序列长度（最大 1024）。

函数返回：

- (2,1)维的矩阵，第 1 行是采样时间矢量，第 2 行为二进制序列矢量 sbpa；
- 高斯白噪声加到二进制序列 varb。

矩阵 sbpa 也保存在 SBPA.mat 文件中，于是它可被 SIMULINK 调用。

gene_sbpa.m file

```
% SBPA.m file generation
% Pseudo-random signal
% l*Te length
% Amplitudes +- A
% Additive Gauss noise b

function [sbpa,varb]=gene_sbpa(A,Te,b,l)
```

```
if nargin~=4
  error('Incorrect arguments number');
end;

n=10;
b=b*randn(1,l);
varb=std(b).^2;
for i=1:l,
   if i<n+1
      sbpa(2,i)=1;
   else
      sbpa(2,i)=-sbpa(2,i-7)*sbpa(2,i-10);
   end;
   sbpa(1,i)=i*Te;
end;
sbpa(2,:)=A.*sbpa(2,:)+b(1,:);

% Sequence recording in a file
save SBPA sbpa;
```

函数 gene_sbpa.m 的检测和结果由文件 test_sbpa.m 得到。

test_sbpa.m file

```
% Pseudo-random generating test
A=1;                    % SBPA Amplitude
Te=0.01;                % SBPA sampling period
b1=0;b2=0.1;b3=0.2;     % Mean square deviation of the noise
N=50;                   % Sequence length (max 1024)

[sbpa1,varb1]=gene_sbpa(A,Te,b1,N);
[sbpa2,varb2]=gene_sbpa(A,Te,b2,N);
[sbpa3,varb3]=gene_sbpa(A,Te,b3,N);

% Representation
figure(1);
subplot(3,1,1),stairs(sbpa1(1,:),sbpa1(2,:));
axis([0 N*Te -2 1.2]);
gtext('Noise variance = 0');
title('Pseudo-random binary sequence');
subplot(3,1,2),stairs(sbpa1(1,:),sbpa2(2,:),'r');
axis([0 N*Te -2.5 2]);
gtext(['Noise variance ='num2str(varb2)]);
ylabel('Amplitude in V');
subplot(3,1,3),stairs(sbpa1(1,:),sbpa3(2,:),'g');
axis([0 N*Te -2.5 2]);
gtext(['Noise variance ='num2str(varb3)]);
xlabel('Time');
```

产生 3 个同样的持续时间为 $50T_s$ 的二进制序列：第 1 个没有累加噪声；第 2 个附加有 0.008 9 方差的噪声；第 3 个附加有 0.039 方差的噪声。

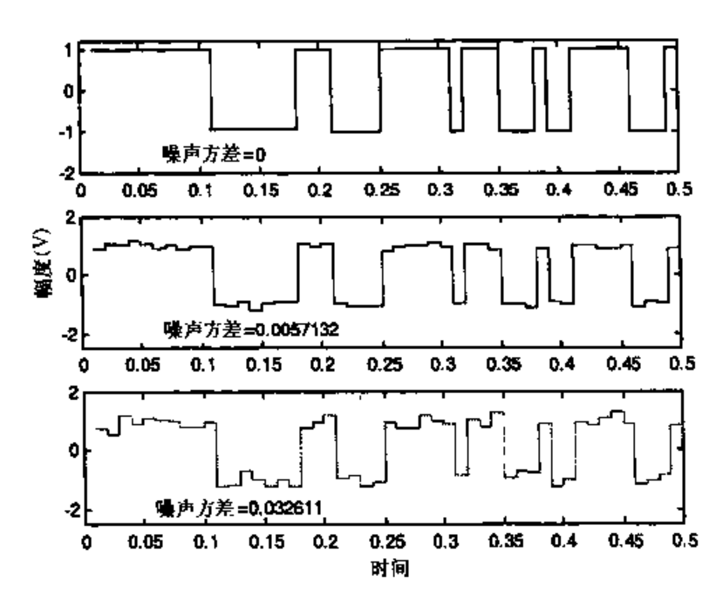


图 9-2　伪随机二进制序列

9.2　色散信道

色散信道呈现对称脉冲响应。它可由一个升余弦函数逼近：

$$h(n)=\frac{1}{2}\left(1+\cos\frac{2\pi}{d}(n-\frac{k+1}{2})\right)$$

其中：

- d　失真系数；
- k　描述脉冲响应的采样数目。

脉冲响应 $h(n)$适用于上述的 n=1~k 的方程，否则为 0。

为模拟具有高斯白噪声的色散信道，给出函数 gene_canal.m，按如下次序传递 4 个参数：

- d　失真系数；
- b　附加高斯白噪声的均方差；
- *sequence*　对应于加在信道输入的序列的矩阵，序列由文件 gene_sbpa.m 产生；
- k　描述信道脉冲响应的采样数目。

函数返回：

- y　在信道输出端得到的对应于序列的线性矢量。

gene_canal.m

```
% Distorsion factor d
% Additive Gauss noise b
% Input vector sequence
% Samples number of the channel impulse response

function [y]=gene_canal(d,b,sequence,k)
if nargin~=4
  error('Incorrect arguments number');
```

```
end;

N=length(sequence);
e=zeros(1,k+1);
b=b*randn(1,N);
% Channel output generation
for j=1:N,
  for i=1:k,
    e(i)=e(i+1);
  end;
  e(k+1)=sequence(2,j);
  y(j)=0;
  for i=1:k,
    y(j)=y(j)+0.5*e(k+1-i)*(1+cos(2*pi*(i-0.5*(k+1))/d));
  end;
  y(j)=y(j)+b(j);
end;
```

函数 gene_canal.m 的检测以及结果由文件 test_canal.m 得到。

test_canal.m file

```
% Transmission channel test
% impulse signal
Ts=0.01;
imp=zeros(2,10);
imp(2,1)=1;
for i=1:10,
  imp(1,i)=i*Ts ;
end;

% Channel output signal noiseless
d1=3.2,d2=3.4,d3=3.6;    % Channel distorsion
b=0,b1=0.1;              % Mean square deviation of the noise
k=5;                     % Samples number
[y1]=gene_canal(d1,b,imp,k);
[y2]=gene_canal(d2,b,imp,k);
[y3]=gene_canal(d3,b,imp,k);
[y4]=gene_canal(d1,b1,imp,k);

% Representation
figure(1)
stairs(imp(1,:),imp(2,:))
axis([Te length(imp)*Te 0 1.1])
Title('Impulse applied to the channel')
xlabel('Time')
ylabel('Amplitude in V')
figure(2)
subplot(3,1,1),stairs(imp(1,:),y1(1,:)),gtext('d = 3.2')
axis([Te length(imp)*Te 0 1.1])
title('Channel impulse response')
subplot(3,1,2),stairs(imp(1,:),y2(1,:),'r:'),gtext('d = 3.4')
axis([Te length(imp)*Te 0 1.1])
```

```
ylabel('Amplitude in V')
subplot(3,1,3),stairs(imp(1,:),y3(1,:),'g-.'),gtext('d = 3.6')
axis([Te length(imp)*Te 0 1.1])
xlabel('Time')
figure(3)
stairs(imp(1,:),y4(1,:)),gtext('Mean square deviation of the noise = 0.1V')
Title('Noised channel impulse response')
xlabel('Time')
ylabel('Amplitude in V')
```

在信道输入处施加幅度为 1 V、T_s=0.01 s 的脉冲，对应于不同的失真系数 d，均有 3 值脉冲响应，而没有噪声加在信道响应上。

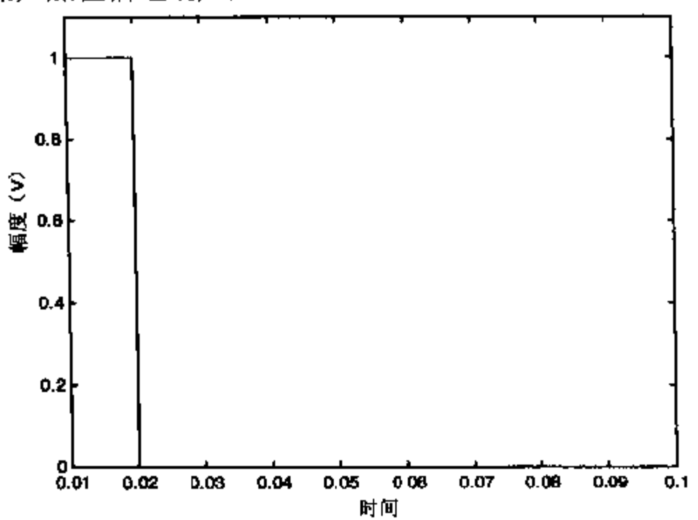


图 9-3　施加在信道上的脉冲

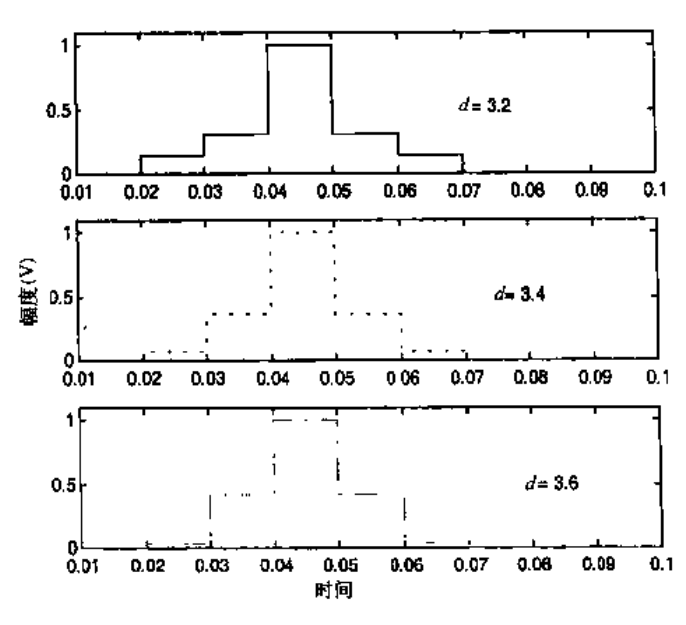


图 9-4　信道脉冲响应

对应于失真系数 d=3.2，具有均方差为 0.1 V 的附加噪声的信道脉冲响应如图 9-5 所示。

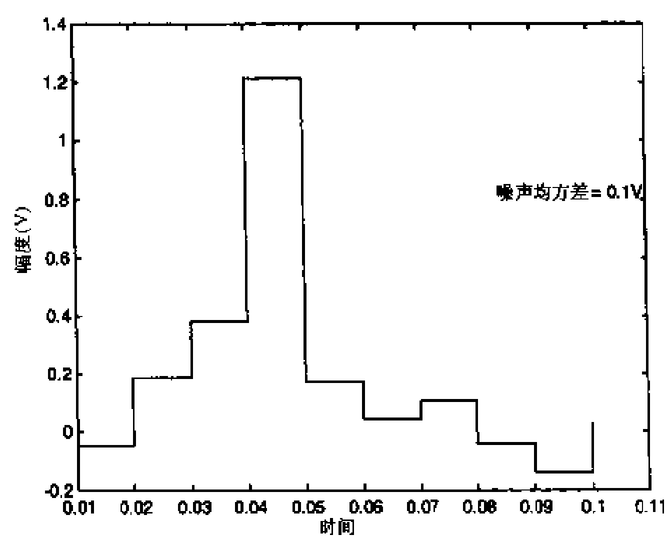


图 9-5　噪声信道脉冲响应

9.3 对称信道均衡器

为了得到自适应均衡器，我们采用 LMS 自适应滤波器。就二进制信号而言，为避免失真，我们感兴趣的是线性相位滤波器。

将持续时间为 T_s 的脉冲加在信道上，信道输出（在均衡器输入处）呈现脉冲发散，最大延迟为 $\frac{k+1}{2}T_s$。如果均衡滤波器是奇数阶 n，为得到对称滤波器，该脉冲与延迟的参考脉冲相比必须延迟 $\frac{1+n}{2}T_s$。与第一个脉冲相比，也就是与在信道输入处所加脉冲相比有 $\left(\frac{k+n}{2}+1\right)T_s$ 的延迟。

具有 n 阶的自适应均衡滤波器在每采样周期上计算：

$$\begin{cases} yf(k)=h(k)x^{\mathrm{T}}(k) \\ \mathrm{err}(k)=e_r(k)-y_f(k) \\ h(k+1)=h(k)+\delta\mathrm{err}(k)x(k) \end{cases}$$

如下文件 test_egalis.m 模拟二进制序列的产生，它通过对称信道和线性相位滤波来传输。

test_egalis.m file

```
% Symmetrial channel equalizer

% Pseudo-random binary sequence noiseless
A=1;             % SBPA Amplitude
Te=0.01;          % SBPA period
b=0;             % Mean square deviation of the noise
N=400;            % Sequence length
[sbpa,varb]=gene_sbpa(A,Te,b,N);

% Representation
figure(1);
```

```
l=100;
stairs(sbpa(1,1:l),sbpa(2,1:l));
axis([0 l*Te -1.2 1.2]);
title('Pseudo-random binary sequence');
xlabel('Time');
ylabel('Amplitude in V');
```

◆ 加在信道上的二进制序列（见图 9-6）

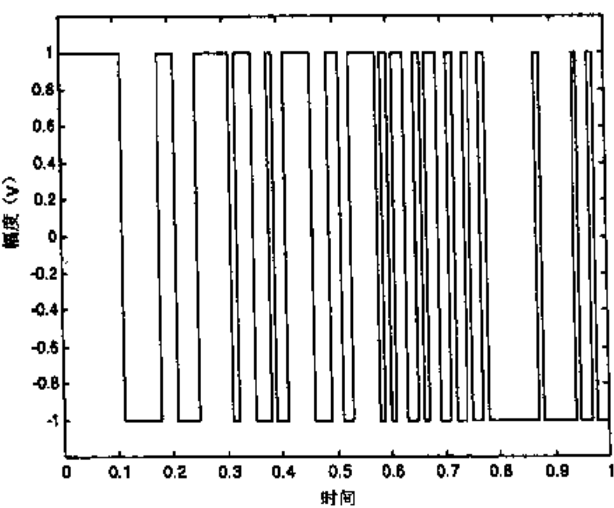


图 9-6　伪随机二进制序列

```
% Noised channel output signal
d=3.2;              % Distorsion channel
b=0.05;             % Mean square deviation of the noise
k=5;                % Samples number
[y]=gene_canal(d,b,sbpa,k);

% Representation
figure(2);
stairs(sbpa(1,1:l),y(1,1:l));
title('Channel output signal');
xlabel('Time');
ylabel('Amplitude in V');
```

◆ 在对称信道输出端得到的序列（见图 9-7）

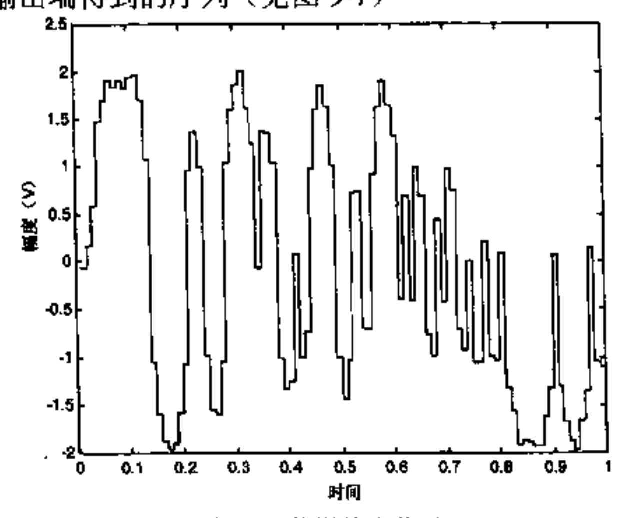


图 9-7　信道输出信号

代表信道脉冲响应的采样数目 k=5，自适应滤波器阶数为 7。因此，使用的参考信号是加在信道输入端、延迟为$\left(\frac{k+n}{2}+1\right)T_s = 7\,T_s$的二进制序列。

```
% LMS adaptive filter

n=7;                % filter order (odd)
delta=0.05;         % adaptation coefficient
x=zeros(1,n);
h=zeros(N+1,n);
yf=zeros(1,(k+n)/2);

for i=1+(k+n)/2:N,
   x(1,n)=y(1,i);
   yf(1,i)=h(i,:)*x';
   err(1,i)=sbpa(2,i-(k+n)/2)-yf(1,i);
   h(i+1,:)=h(i,:)+delta*err(1,i)*x;
   for j=1:n-1,
      x(1,j)=x(1,j+1);
   end;
end;

% Results representation

figure(3)
% Input channel binary sequence
subplot(2,1,1)
stairs(sbpa(1,N-1:N),sbpa(2,N-1-(k+n)/2:N-(k+n)/2),'r')
axis([(N-1)*Te N*Te -1.2 1.2])
Title('Channel input and filter output')
% Filter output binary sequence
subplot(2,1,2)
stairs(sbpa(1,N-1:N),yf(1,N-1:N))
xlabel('Time')
```

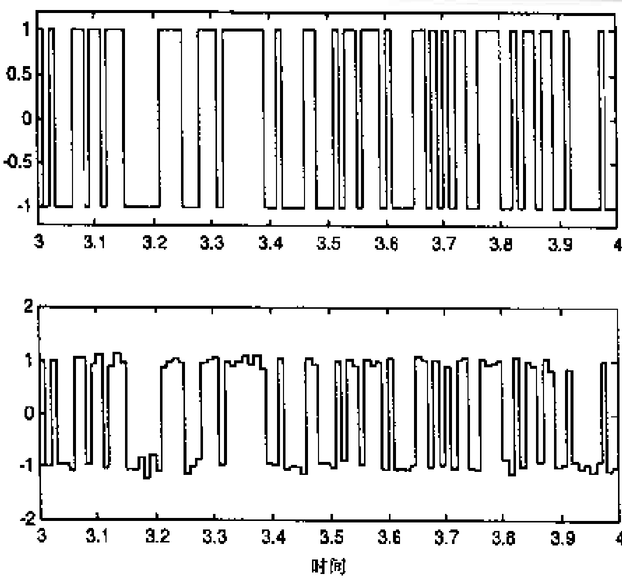


图 9-8　信道输入和滤波器输出

滤波后的序列很有用且无任何误差。作为最佳判定标准，为了返回发送的二进制序列，只需采用滤波器输出信号符号。

```
% Filter impulse response
figure(4)
stem(h(N+1,:))
line([1 7],[0 0])
title('Filter impulse response')
xlabel('Coefficients')
```

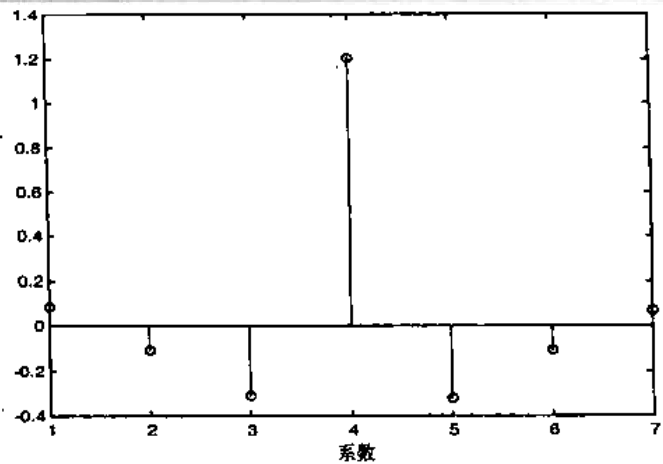


图 9-9　滤波器脉冲响应

由经过 400 次迭代后得到的滤波器系数确实可得到一个线性相位对称滤波器。

```
% Coefficients evolution

figure(5)
plot(h)
title('Coefficients evolution')
xlabel('Samples')
gtext('h3'),gtext('h0=h6'),gtext('h1=h5')
gtext('h2=h4')
```

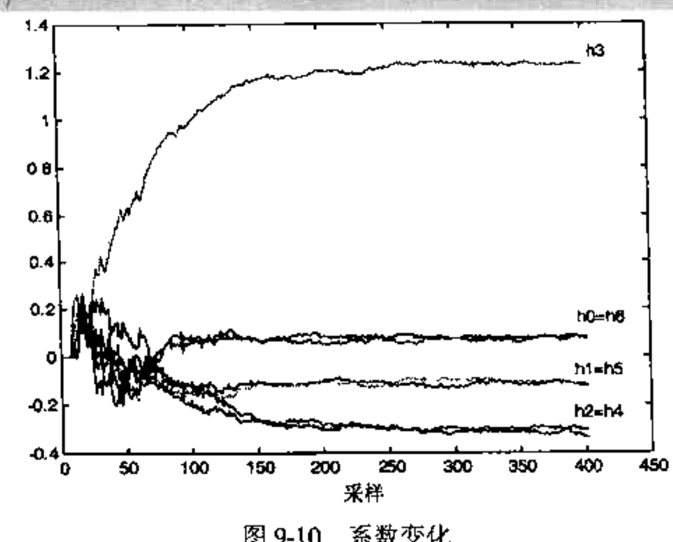


图 9-10　系数变化

约 150 次迭代后，滤波器系数趋于稳定。同时可以看到，这里的系数是对称的。

```
% Square error evolution
figure(6)
semilogy(sbpa(1,1+(k+n)/2:N),err(1,1+(k+n)/2:N).^2)
title('Square error evolution')
xlabel('Time')
```

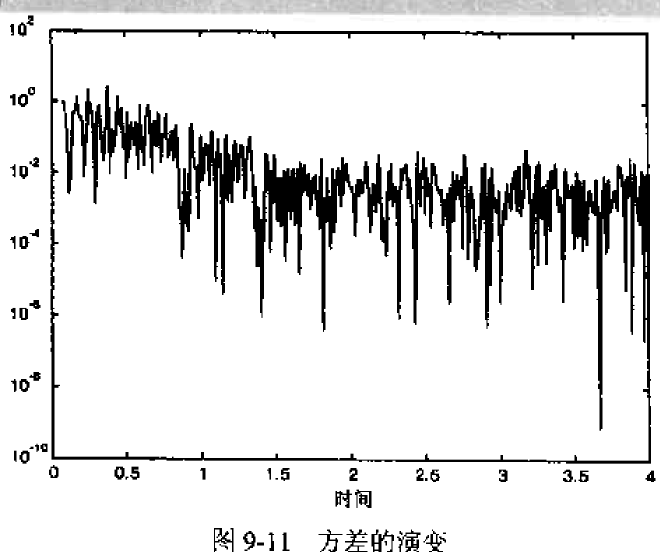


图 9-11　方差的演变

现在假定再次在如前述同样的信道上施加同样的二进制序列，但是为非对称自适应滤波器。滤波器阶数保持为 7，但是参考输入信号与加在信道上的信号相比仅有 $3T_e$ 的延迟，这是为了能够与信道输出的最大幅度采样进行比较。

test_egalisl.m file

```
% Symmetrical channel equalizer

% Pseudo random binary sequence noiseless
A=1;                % SBPA amplitude
Te=0.01;            % SBPA period
b=0;                % Mean square deviation of the noise
N=1000;             % Sequence length
[sbpa,varb]=gene_sbpa(A,Te,b,N);

% Output signal noised channel
d=3.2;              % Distortion channel
b=0.05;             % Mean square deviation of the noise
k=5;                % Samples number (odd)
[y]=gene_canal(d,b,sbpa,k)

% LMS adaptive filter
n=7;                % filter order (odd) for symmetrical
delta=0.01;         % Adaptation coefficients
x=zeros(1,n);
h=zeros(N+1,n);
yf=zeros(1,(k+1)/2);
for i=1+(k+1)/2:N,
   x(1,n)=y(1,i);
   yf(1,i)=h(i,:)*x';
   err(1,i)=sbpa(2,i-(k+1)/2)-yf(1,i);
   h(i+1,:)=h(i,:)+delta*err(1,i)*x;
```

```
    for j=1:n-1,
        x(1,j)=x(1,j+1);
    end;
end;

% Results representation
l = 100;
figure(3)
% Channel input binary sequence
subplot(2,1,1)
stairs(sbpa(1,N-l:N),sbpa(2,N-l-(k+1)/2:N-(k+1)/2),'r')
axis([(N-l)*Te N*Te -1.2 1.2])
Title('Channel input and filter output')
% Filter output binary sequence
subplot(2,1,2)
stairs(sbpa(1,N-l:N),yf(1,N-l:N))
xlabel('Time')

% Filter impulse response
figure(4)
handle = stem(h(N+1,:)); set(handle,'LineWidth',2), hold on
line([1 7],[0 0])
title('Filter impulse response')
xlabel('Coefficients')

% Coefficients evolution
figure(5)
plot(h)
title('Coefficients evolution')
xlabel('Samples')
% Square error evolution
figure(6)
semilogy(sbpa(1,1+(k+n)/2:N),err(1,1+(k+n)/2:N).^2)
title('Square error evolution')
xlabel('Time')
```

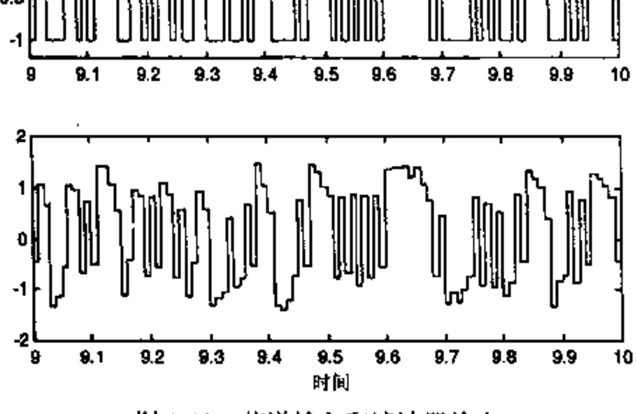


图 9-12　信道输入和滤波器输出

滤波后的信号在判定后仍然有用，尽管它与对称滤波器输出的结果相比质量较差。

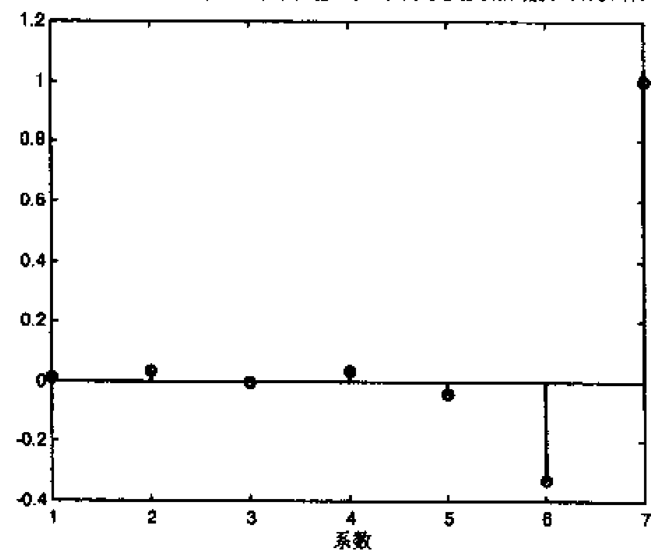


图 9-13　滤波器脉冲响应

从逻辑上看，得到的滤波器是完全非对称的。

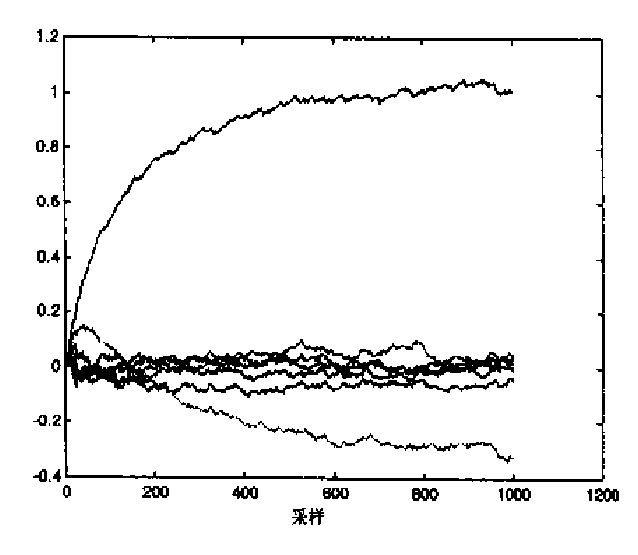


图 9-14　系数演变

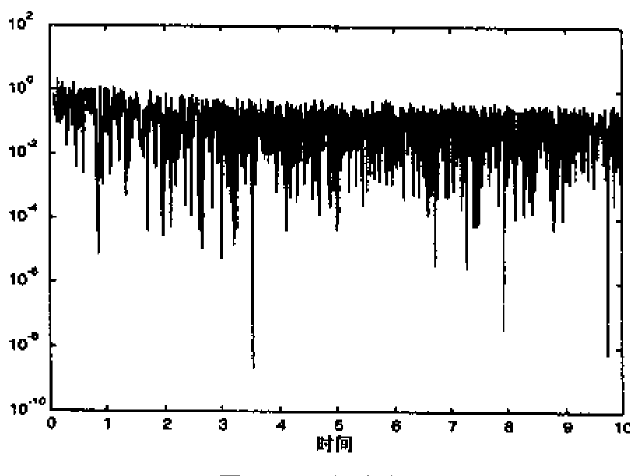


图 9-15　方差演变

比较误差变化图，可以很清楚地看出对称滤波器的优越性。

9.4 使用 SIMULINK

使用 SIMULINK 的信道均衡需要使用 2 个 S 函数块，第 1 个块用于对称信道的模型，第 2 个块用于自适应滤波器模型。加在信道上的二进制序列由块“From file”产生，并受由函数 gene_sbpa.m 产生的文件 SBPA.mat 的影响，如图 9-16 表示。

因为它要成为自适应滤波器参考信号，所以加到二进制序列中的延迟在产生固定延迟的块“Transport delay”帮助下得到。

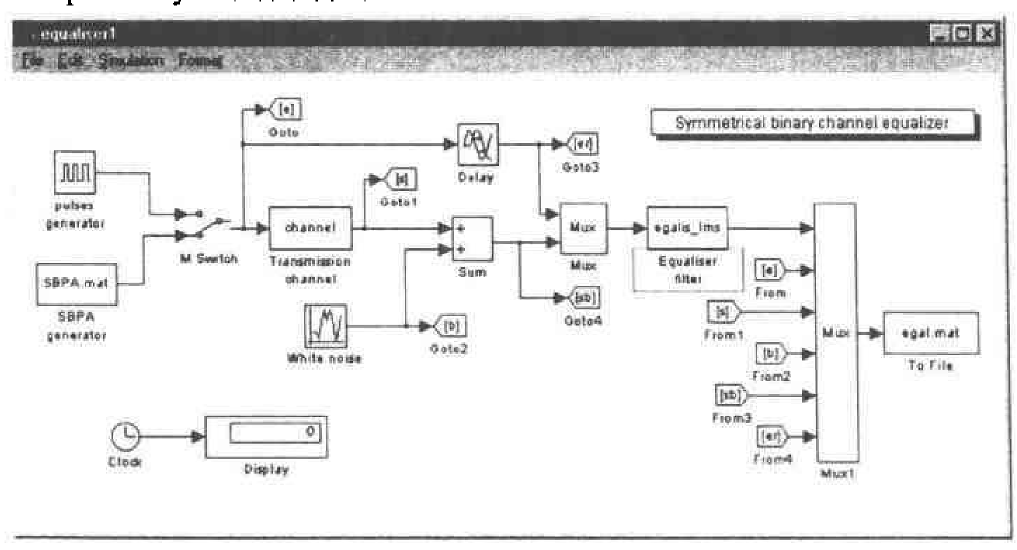


图 9-16　对称二进制信道均衡器

9.4.1　S 函数，传输信道

除了系统参数，仿真传输信道的 S 函数接收到如下输入参数：

- T_s　采样周期；
- d　失真系数；
- k　信道脉冲响应的采样数目。

得到的一个输入和一个输出分别代表对称信道的输入和输出。

Channel.m file

```
function [sys,x0,str,ts]=Channel(t,x,u,flag,Ts,d,k)

global  e

switch flag,
case 0          % Initialization stage
  e=zeros(1,k+1);
  [sys,x0,str,ts]=Initialization(Ts);

case 3          % Output stage
  for i=1:k,
    e(1,i)=e(1,i+1);
```

```
  end;
  e(k+1)=u(1);
  y=0;
  for i=1:k,
     y=y+0.5*e(1,k+1-i)*(1+cos(2*pi*(i-(k+1)/2)/d));
  end;
  sys=y;
case {1,2,4,9}  % Not used stages
  sys=[];
otherwise
  error(['Unhandled flag=',num2str(flag)]);
end

% Initialization function
  function [sys,x0,str,ts]=Initialization(Ts)
  sizes=simsizes;
  sizes.NumContStates=0;
  sizes.NumDiscStates=0;
  sizes.NumOutputs=1;
  sizes.NumInputs=1;
  sizes.DirFeedthrough=0;
  sizes.NumSampleTimes=1;
  sys=simsizes(sizes);
  x0=[];
  str=[];
  ts=[Ts 0];
```

9.4.2 S 函数，lms 型自适应均衡器

除了系统参数，仿真自适应均衡器的 S 函数接收到如下输入参数：

- T_s 采样周期；
- *delta* 自适应增益；
- *n* 滤波器阶数。

它包括 2 个输入：参考输入 $u(1)=e_r(t)$和滤波器输入 $u(2)=sb(t)$；2 个输出分别为滤波器输出 $y_f(t)$和误差输出 $err(t)$。

egalis_lms.m file

```
function [sys,x0,str,ts]=lms2(t,x,u,flag,Ts,delta,n)
global err h xe yf coefs i

switch flag,
case 0
   clear coefs;
   i=1;
   xe=zeros(1,n);
   h=zeros(1,n);
   err=0;
   [sys,x0,str,ts]=Initialization(Ts);
case 3
   xe(n)=u(2);
```

```
  yf=h*xe';
  err=u(1)-yf;
  h=h+delta*err*xe;
  for j=1:n-1
    xe(j)=xe(j+1);
  end
  coefs(i,1:n)=h(1:n);
  sys=[yf err];
  i=i+1;
case 9
  save coefs coefs;
case {1,2,4}
  sys=[];
otherwise
  error(['Unhandled flag=',num2str(flag)]);
end
  function [sys,x0,str,ts]=Initialization(Ts)
  sizes=simsizes;
  sizes.NumContStates=0;
  sizes.NumDiscStates=0;
  sizes.NumOutputs=2;
  sizes.NumInputs=2;
  sizes.DirFeedthrough=0;
  sizes.NumSampleTimes=1;
  sys=simsizes(sizes);
  x0=[];
  str=[];
  ts=[Ts 0];
```

9.4.3 仿真结果

得到的仿真结果包含如下参数的仿真：

- T_s=0.01s　采样周期；
- d=3.2　信道失真系数；
- e_c=0.01V　附加高斯的噪声均方差；
- k=5　信道脉冲响应的采样数目；
- $delta$ = 0.05　LMS 自适应均衡器的自适应系数；
- N=7　自适应均衡滤波器阶数。

为得到对称滤波器需选择延迟、文件 egalis1.m 给出具有 4 s 持续时间的仿真结果。

egalis1.m file

```
% Channel equalizer
% Results reading
load egal.mat
t=signaux(1,:);
yf=signaux(2,:);
err=signaux(3,:);
e=signaux(4,:);
s=signaux(5,:);
b=signaux(6,:);
```

```
sb=signaux(7,:);
er=signaux(8,:);

% Signals representation
figure(1)
N=length(t);
l=100;
Ts=0.01;
% Input channel binary sequence
subplot(5,1,1)
stairs(t(N-l:N),e(N-l:N))
axis([(N-l-1)*Ts (N-1)*Ts -1.2 2])
Title('Binary sequences')
% Output binary sequence
subplot(5,1,2)
stairs(t(N-l:N),s(N-l:N),'r')
axis([(N-l-1)*Ts (N-1)*Ts -2 3])
% Channel+noise binary sequence
subplot(5,1,3)
stairs(t(N-l:N),sb(N-l:N),'r')
% Reference binary sequence
subplot(5,1,4)
stairs(t(N-l:N),er(N-l:N))
axis([(N-l-1)*Ts (N-1)*Ts -1.2 2])
% Output filter binary sequence
subplot(5,1,5)
stairs(t(N-l:N),yf(N-l:N))
axis([(N-l-1)*Ts (N-1)*Ts -2 3])
xlabel('Time')
gtext('Channel input')
gtext('Channel output')
gtext('Output noised channel')
gtext('Reference input')
gtext('Filter output')

load coefs.mat
figure(2)
plot(coefs)
title('Filter coefficients evolution')
xlabel('Samples')
figure(3)
stem(coefs(N,:)), hold on
line([1 7],[0 0]), hold off
title('Filter impulse response')
xlabel('Coefficients')
figure(4)
semilogy(t,err.^2)
title('Square error evolution')
xlabel('Time')
```

用具有零阈值的滤波器二进制序列进行最佳判定后，接收序列与发送序列完全相同（见

图 9-17)。

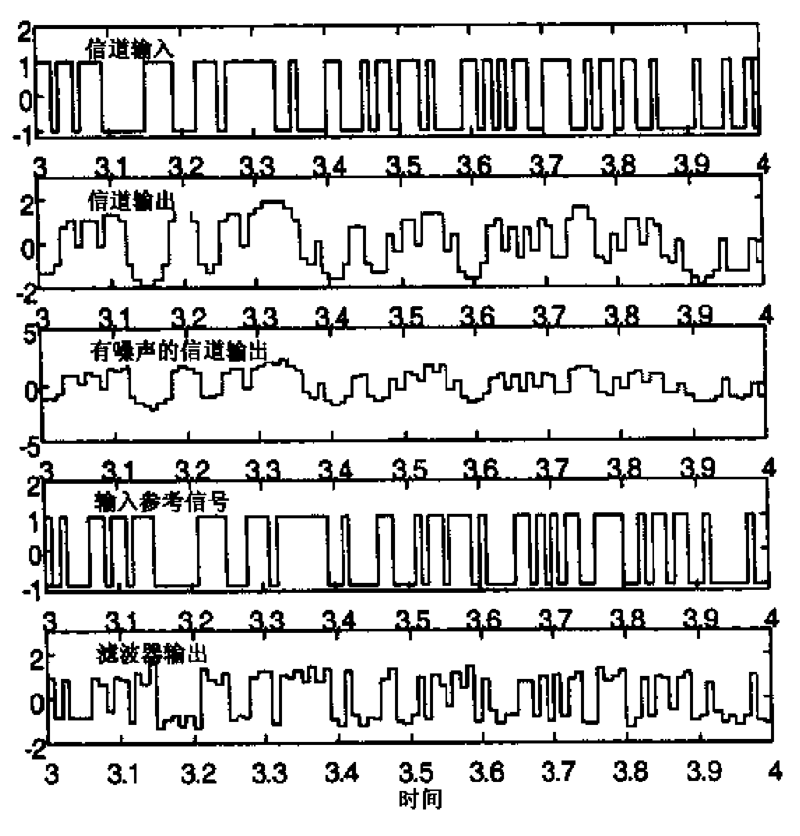


图 9-17 二进制序列

因为滤波器阶数为 7，所以系数向中心在 h3 的对称滤波器变化（见图 9-18)。

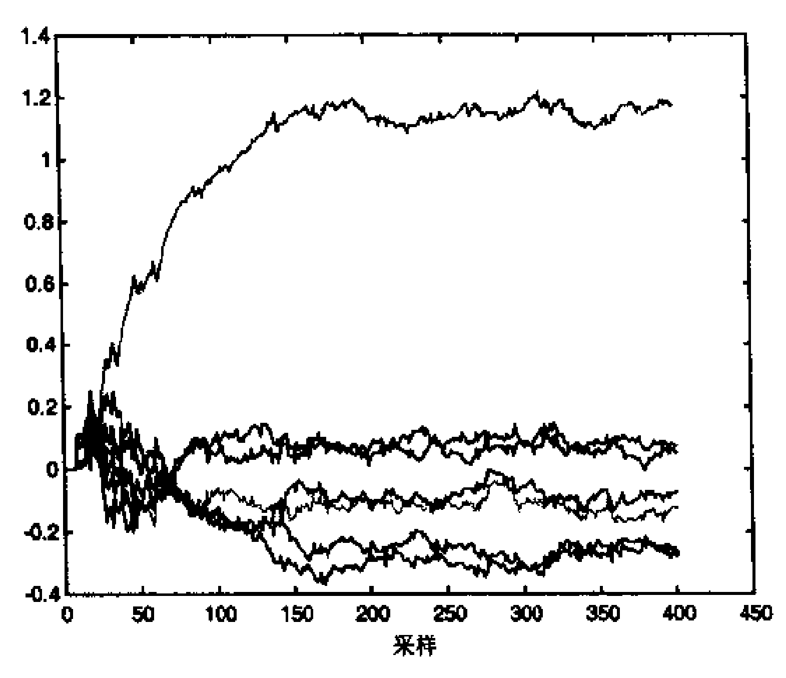


图 9-18 滤波器系数趋势图

◆ 400 次迭代后得到的滤波器系数（见图 9-19)

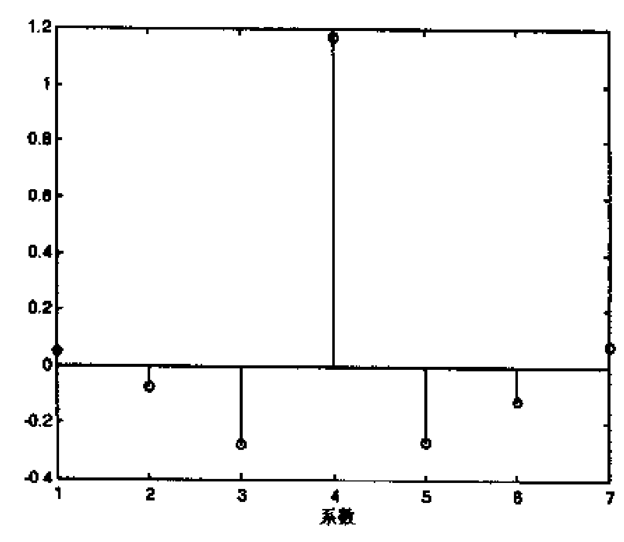


图 9-19　滤波器脉冲响应

◆　方差变化（见图 9-20）

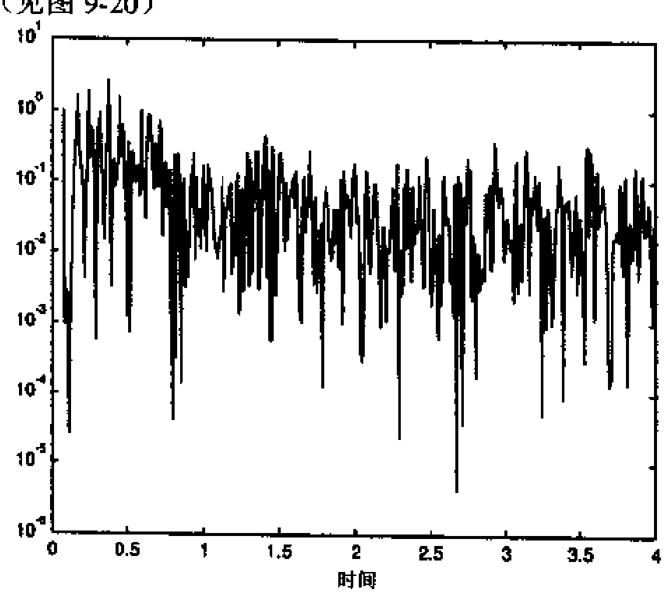


图 9-20　方差变化图

附录A　SIMULINK 3 的 S 函数

A.1　SIMULINK 3 的 S 函数功能原理

S 函数使用特定的语法使动态系统具有交互功能。它们以连续、离散或混合方式最大程度地使自身与系统相适应。

S 函数特别与下列情况有关系：

- 增加一个模块函数至 SIMULINK；
- 将一部分 C 语言程序代码加入模型中；
- 以数学方程形式描述系统；
- 使用图形制作。

每个 SIMULINK 块一般具有如下特性：

- 输入矢量 u；
- 输出矢量 y；
- 状态矢量 x。

这些不同矢量之间的关系用如下等式来描述：

$$y = f_0(t,x,u)$$
$$\dot{x}_c = f_d(t,x,u)$$
$$x_{k+1} = f_u(t,x,u)$$

其中

$$x = \begin{bmatrix} x_c \\ x_k \end{bmatrix}$$

状态矢量 x 由两部分组成：第一部分代表连续状态；第二部分代表离散状态。

A.2　仿真的不同阶段

在仿真过程中，SIMULINK 在矢量更新阶段产生重复调用，在仿真开始的初始阶段和结束停止阶段都伴随着这一情况。

标识器 flag 使得在任何时候都知道系统处于哪个仿真阶段。

S 函数可写为 M 文件或 C-MEX 文件。

标识器 flag 的含义

模 拟 阶 段	M 文件标识阶段	调用“C-MEX file”函数
初始化	0	MdlInitializeSizes
下一采样计算（可选）	4	mdlGetTimeOfNextVarHit
输出计算	3	mdlOutputs
离散状态更新	2	mdlUpdate
连续状态更新	1	mdlDerivatives
仿真结束	9	mdlTerminate

A.3 通过 M 文件调用产生 S 函数

由 S 函数语法指定的 M 文件如下：

```
function[sys,x0,str,ts]=function_name(t,x,u,flag,p1,p2,…)
```

t,x,u,flag：通过系统赋给 S 函数的默认驱动变量；

p1，p2：希望赋给 S 函数的可选变量。

根据时间，返回系统的变量依赖矢量 x、u 和标识器 flag。

- flag=0，初始函数必须返回 sys（系统大小信息），返回 x0（系统初始状态）；返回至 *str* 一空链（对函数 m）；采样瞬间至 *ts*。为此，可使用函数 mdlInitializeSizes 使得包含如下影响变量 *sys* 范围的结构 simsize-type 初始化。

```
NumContStates           % 连续状态数目
NumDiscStates           % 离散状态数目
NumOutputs              % 输出数目
NumInputs               % 输入数目
DirFeedthrough          % D空矩阵
NumSampleTimes          % 矩阵 ts 行数目
```

T_s 是包含采样瞬间[period offset]的[NumSampleTimes,2]矩阵。

- flag=1，连续状态更新函数必须将 $\dot{x}$ 返回 *sys*；
- flag=2，离散状态更新函数必须将 $x(k+1)$ 返回 *sys*；
- flag=3，输出更新函数必须将 y 返回 *sys*；
- flag=4，在采样步变化的情况下，下一个采样瞬间返回 *sys*；
- flag=9，结束仿真，系统变量消失，*sys* 返回一个(*sys*=[ ])的空矢量。

◆ 例 1　用 S 函数仿真连续状态系统

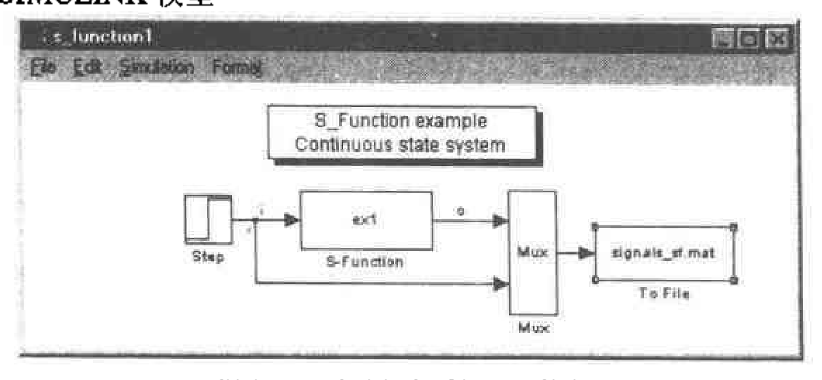
我们用 S 函数 ex1.m 来仿真单输入和单输出的连续过程，状态表示为：

$$\boldsymbol{A}=\begin{bmatrix}-1.5 & -1.25\\ 4 & 0\end{bmatrix} \qquad \boldsymbol{B}=\begin{bmatrix}2\\0\end{bmatrix}$$

$$\boldsymbol{C}=\begin{bmatrix}0 & 1.5625\end{bmatrix} \qquad \boldsymbol{D}=0$$

在没有其他参数传递的情况下调用 S 函数，施加单位阶跃形式的输入信号。

◆ 编辑 SIMULINK 模型



附图 1-1　连续状态系统 S 函数举例

文件 ex1.m 对每个使用的事件进行标志检测并调用函数。

- flag=0，初始化函数 S 函数的参数调用：
  - 连续状态数目；
  - 离散状态数目；
  - 输入数目；
  - 输出数目；
  - 采样瞬间数目；
  - 初始状态；
  - 采样消除和时间。

当使用矩阵 ***D*** 或 flag=4 时，初始化结构的 DirFeedthrough 域放在位置 1。

对于连续过程[0 0]，维数为（1,2）的矢量 *ts*，采样误差和周期为 0。对于用一可变采样（用于 flag=4）的离散过程，这个矢量必须初始化为[-2 0]。

- flag=1，调用连续状态计算函数：

$$x = Ax + Bu$$

- flag=3，调用输出计算函数：

$$y = Cx + Du$$

ex1.m file

```
% Continuous state model calculation

function [sys,x0,str,ts] = ex1(t,x,u,flag)
% Continuous state model
  A = [-1.5 -1.25 ; 4 0];
  B = [2 ; 0];
  C = [0 1.5625];
  D = 0;

switch flag

case 0  % Initialization stage
  [sys,x0,str,ts] = Initialization(A,B,C,D);

case 1  % State calculation stage
  sys = DerivatesCalcul (x,u,A,B);

case 3  % Output calculation stage
  sys = OutputsCalcul(x,u,C,D);

case {2,4,9}    % unused stages
  sys = [];

otherwise
  error(['Unhandled flag = ',num2str(flag)]);
end

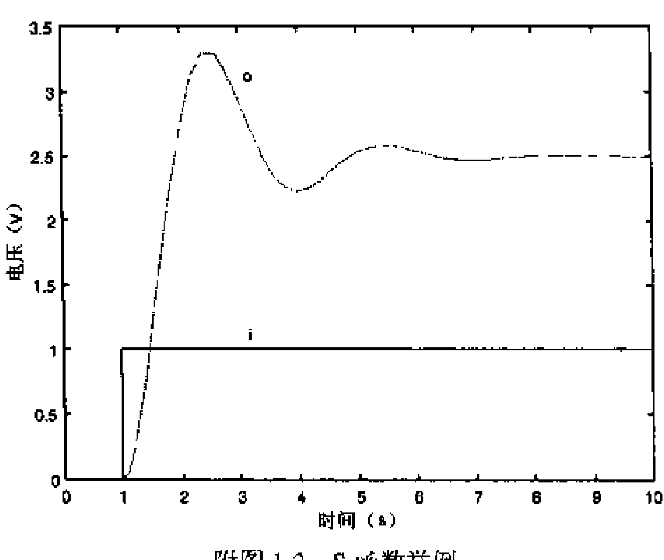
% Initialization function
function [sys,x0,str,ts] = Initialization(A,B,C,D)
```

```
if D ==0
  DirF = 0;
else
  DirF = 1;
end
 sizes = simsizes;              % simsiezs type structure
 sizes.NumContStates = length(A); % continuous states number
 sizes.NumDiscStates = 0;         % discretes states number
 sizes.NumOutputs = 1;            % outputs number
 sizes.NumInputs = 1;             % inputs number
 sizes.DirFeedthrough = DirF;     % 0 if D = 0, 1 if D ¹ 0
                                % of if flag = 4 used
 sizes.NumSampleTimes = 1;  % sampling instants number
 sys = simsizes(sizes);
 x0 = zeros(size(B));
 str = [];
 ts = [0 0];     % continuous process

% State calculation function
function sys = DerivatesCalcul (x,u,A,B)
sys = A*x+B*u;

% Output calculation function
function sys = OutputsCalcul (x,u,C,D)
sys = C*x+D*u;
```

◆ 仿真结果



附图 1-2　S 函数举例

◆ 例 2　用 S 函数仿真一个离散状态模型

在 ex2.m 文件中，我们用 S 函数来仿真前面采样周期 T_c 为 0.1 s 的离散化过程的例子。

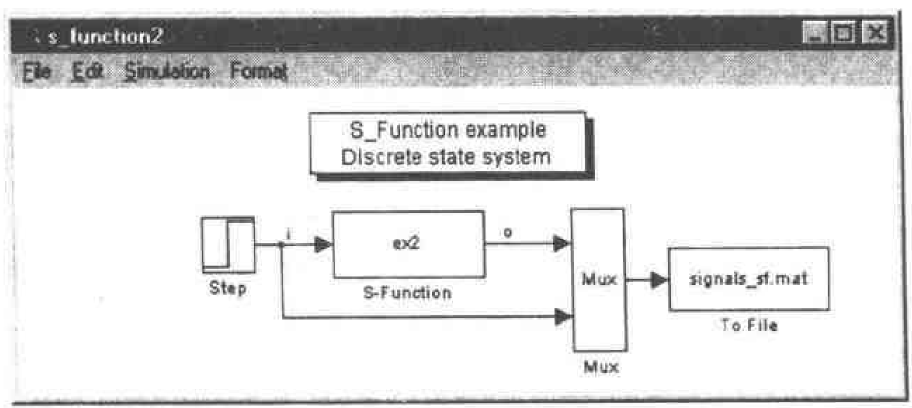
$$\boldsymbol{A}_d=\begin{bmatrix}1.815 & -0.86\\ 1 & 0\end{bmatrix}\qquad \boldsymbol{B}_d=\begin{bmatrix}0.25\\ 0\end{bmatrix}$$

$$\boldsymbol{C}_d=\begin{bmatrix}0.237 & 0.225\end{bmatrix}\qquad \boldsymbol{D}_d=0$$

在另外 2 个参数 T_e 和 X_i 传递的情况下调用 S 函数，这 2 个参数分别代表采样周期和

状态初始条件。施加一个单位阶跃型输入信号。

◆ SIMULINK 模型编辑



附图 1-3 离散状态系统 S 函数举例

flag=1 和 flag=2 的情况分别用于离散和连续过程。

ex2.m file

```
% Discrete state model simulation
function [sys,x0,str,ts] = ex2(t,x,u,flag,Te,Xi)
  Ad = [1.815 -0.86 ; 1 0];
  Bd = [0.25 ; 0];
  Cd = [0.237 0.225];
  Dd = 0;

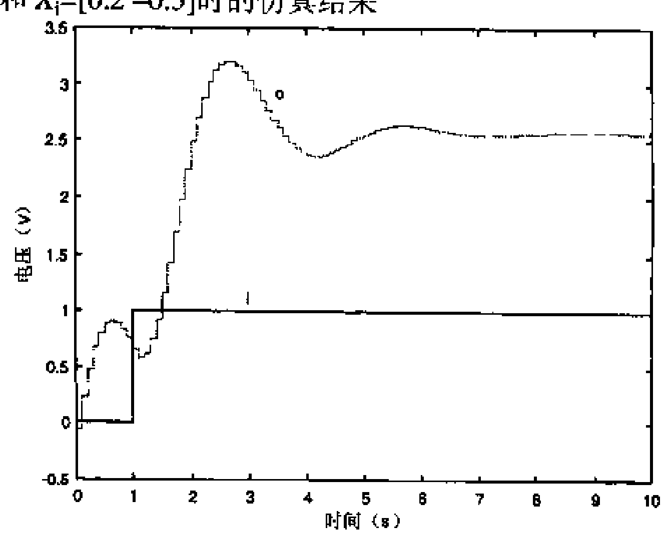
switch flag,

case 0   % Initialization stage
  [sys,x0,str,ts] = Initialization(Ad,Dd,Te,Xi);
case 2   % State calculation stage
  sys = DerivatesCalcul (x,u,Ad,Bd);
case 3   % Output calculation stage
  sys = OutputsCalcul (x,u,Cd,Dd);
case {1,4,9}   % Unused stages
  sys = [];
otherwise
  error(['Unhandled flag = ',num2str(flag)]);
end
% Initialization function
function [sys,x0,str,ts] = Initialization(Ad,Dd,Te,Xi)
if Dd ==0
  DirF = 0;
else
  DirF = 1;
end;
  sizes = simsizes;
  sizes.NumContStates = 0;
  sizes.NumDiscStates = length(Ad);
  sizes.NumOutputs = 1;
  sizes.NumInputs = 1;
```

```
  sizes.DirFeedthrough = DirF;
  sizes.NumSampleTimes = 1;
  sys = simsizes(sizes);
  x0 = Xi;      % On-state original conditions
  str = [];
  ts = [Te 0]; % offset less Te sampling period

% State calcuation function
function sys = DerivatesCalcul (x,u,Ad,Bd)
sys = Ad*x+Bd*u;
% Output calculation function
function sys = OutputsCalcul (x,u,Cd,Dd)
sys = Cd*x+Dd*u;
```

◆ 在 T_s=0.1s 和 X_i=[0.2 –0.5]时的仿真结果



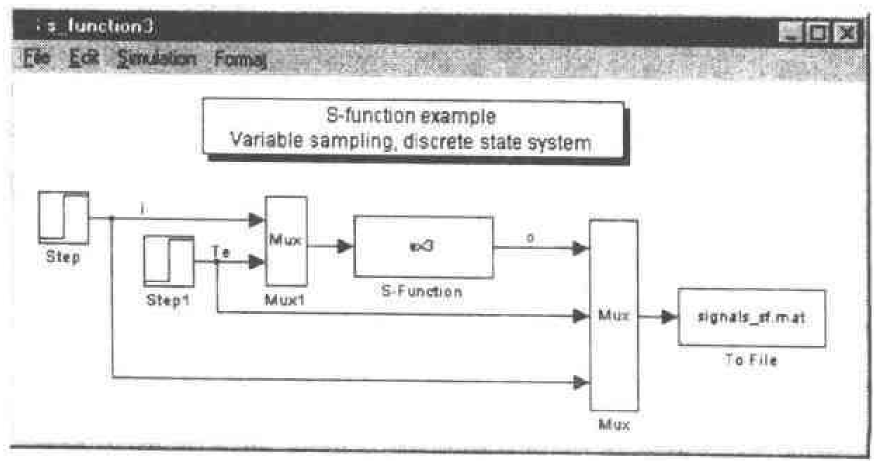
附图 1-4 S 函数举例

◆ 例 3 用 S 函数仿真变采样周期的离散状态模型

用 S 函数 ex3.m 仿真前述例子的离散过程，对比过程加一输入以确定过程采样周期。

调用表示状态初始情况的附加参数 X_i 的 S 函数。采样周期（以 s 计量）等于电压幅度（以 V 计量），并施加在 S 函数第 2 个输入上。

◆ SIMULINK 模型编辑



附图 1-5 离散状态系统变化采样的 S 函数举例

flag=4 时用于未来采样瞬间的计算。为此，有必要使 S 函数初始结构元素 DirFeedThrough 置 1，置初始化 ts=[-2 0]。

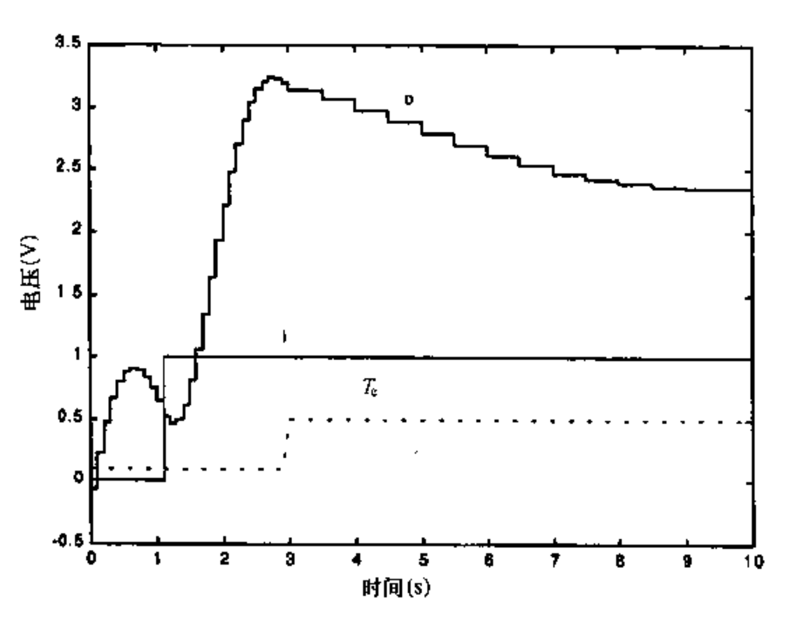
ex3.m file

```
% Discrete state model simulation
% Variable sampling period
function [sys,x0,str,ts] = ex3(t,x,u,flag,Xi)
  Ad = [1.815 -0.86 ; 1 0]; Bd = [0.25 ; 0];
  Cd = [0.237 0.225]; Dd = 0;

switch flag
case 0       % Initialization stage
  [sys,x0,str,ts] = Initialization(Ad,Xi);
case 2       % State calculation stage
  sys = DerivatesCalcul (x,u,Ad,Bd);
case 3       % Output calculation stage
  sys = OutputsCalcul (x,u,Cd,Dd);
case 4       % Next Te calculation stage
  sys = NextTeCalcul (t,u);
case {1,9}  % Unused stages
  sys = [];
otherwise
  error(['Unhandled flag=',num2str(flag)]);
end

% Initialization function
function [sys,x0,str,ts] = Initialization(Ad,Xi)
 sizes = simsizes;
 sizes.NumContStates = 0;
 sizes.NumDiscStates = length(Ad);
 sizes.NumOutputs = 1;
 sizes.NumInputs = 2;
 sizes.DirFeedthrough = 1;  % The flag=4 use requires to
    % position DirFeedthrough=1
 sizes.NumSampleTimes = 1;
 sys = simsizes(sizes);
 x0 = Xi;
 str = [];
 ts = [-2 0];       % In order to obtain a sampling  variable step
% State calculation function
function sys = DerivatesCalcul (x,u,Ad,Bd)
sys = Ad*x+Bd*u(1);
% Output calculation function
function sys = OutputsCalcul (x,u,Cd,Dd)
sys = Cd*x+Dd*u(1);
% Next Te function calculation
function sys = NextTeCalcul (t,u)
sys = t+u(2);
```

◆ 在 X_i=[0.2 -0.5]，T_e=0.1 s，t=0~3 s；T_e=0.5 s，t>3 s 时的仿真结果（见附图 1-6）



附图 1-6　S 函数电压举例

A.4　通过 C MEX 文件调用产生 S 函数

采用以前的例子，将其描述成 C MEX。当然有必要配置 C 或 C++编译器。在无其他参数传递下调用函数，初始状态为 0，采样周期为 0.1 s。标识器 flag 通过调用函数而被替代。函数 mdlUpdate、mdlDerivatives 和 mdlTerminate 分别对应于 M 函数在 flag=2、flag=1 和 flag=9 的情况，即使没使用也必须显示。本情况下，函数代码为空。

ex4.c file

```
/* Name definition of the S-function */
#define S_FUNCTION_NAME ex4

/* Necessary inclusion to the SimStruct definition and of
the associated macros */
#include "simstruc.h"

/* State matrix declaration */
static double Ad[2][2] = {{1.815, -0.86},{1 ,0}};
static double Bd[2][1] = {{0.25},{0}};
static double Cd[1][2] = {0.237, 0.225};
static double Dd = 0;

/* S-function 's characteristics ' initialization */
static void mdlInitializeSizes(SimStruct *S)
{
  ssSetNumContStates(S,0);      /* Continuous states number */
  ssSetNumDiscStates(S,2);      /* Discrete states number    */
  ssSetNumInputs(S,1);         /* Inputs number             */
  ssSetNumOutputs(S,1);        /* Outputs number            */
  ssSetDirectFeedThrough(S,0); /* direct feedthrough flag */
  ssSetNumSampleTimes(S,1);     /* sampling instants number */
```

```
  ssSetNumSFcnParams(S,0);
  ssSetNumRWork(S,0);
  ssSetNumIWork(S,0);
  ssSetNumPWork(S,0);
}
/* Offset less 0.1s sampling period initialization */
static void mdlInitializeSampleTimes(SimStruct *S)
{
ssSetSampleTime(S, 0, 0.1);
  ssSetOffsetTime(S, 0, 0.0);
}
/* Nil initial conditions  initialization */
static void mdlInitializeConditions(real_T *x0, SimStruct *S)
{
int i;
for (i = 0; i<2; i++)
 {
  x0++ = 0;
 }
}

/* Output calculation */
static void mdlOutputs(real_T *y, const real_T *x,
const real_T *u, SimStruct *S, int_T tid)
{
  y[0] = Cd[0][0]*x[0]+Cd[0][1]*x[1];
}

/* Discrete state updating */
static void mdlUpdate(real_T *x, const real_T *u,
SimStruct *S, int_T tid)
{
 double varX[2];
 varX[0] = Ad[0][0]*x[0]+Ad[0][1]*x[1]+Bd[0][0]*u[0];
 varX[1] = Ad[1][0]*x[0]+Ad[1][1]*x[1]+Bd[1][0]*u[0];
 x[0] = varX[0];
 x[1] = varX[1];
}

/* Continuous stage updating*/
static void mdlDerivatives(real_T *dx, const real_T *x,
const real_T *u, SimStruct *S, int_T tid)
{
/* Absence of code because the function is discrete */
}

static void mdlTerminate(SimStruct *S)
{
/* Absence of code because the function is not used in this case */
}
```

```
/* Necessary heading for the S-function s */
#ifdef MATLAB_MEX_FILE
#include "SIMULINK.c"
#else
#include "cg_sfun.h"
#endif
```

为编辑本文件，有必要确定编译器类型及其安装路径。为此，必须执行命令 cmex-setup 并回答问题。接着，产生文件 mexopts.bta，包含路径、命令和编译选项及连接。然后在目录<MATLAB>\BIN 下在文件 cmex.bat 的帮助下就可编辑.C 文件，其中<MATLAB>指定 MATLAB 的安装路径。

附录 B 在 SIMULINK 3 中对一组块进行封装

封装或屏蔽一组像 S 函数这样的块是 SIMULINK 很重要的功能。封装一组块只是为了完成一个特定任务（如计算、信号发生等）。在这些封装中，可以看到：

- 连接每个用户域的特定模块的产生；
- 对话框数目减少；
- 组织分层结构，简化 SIMULINK 模型。

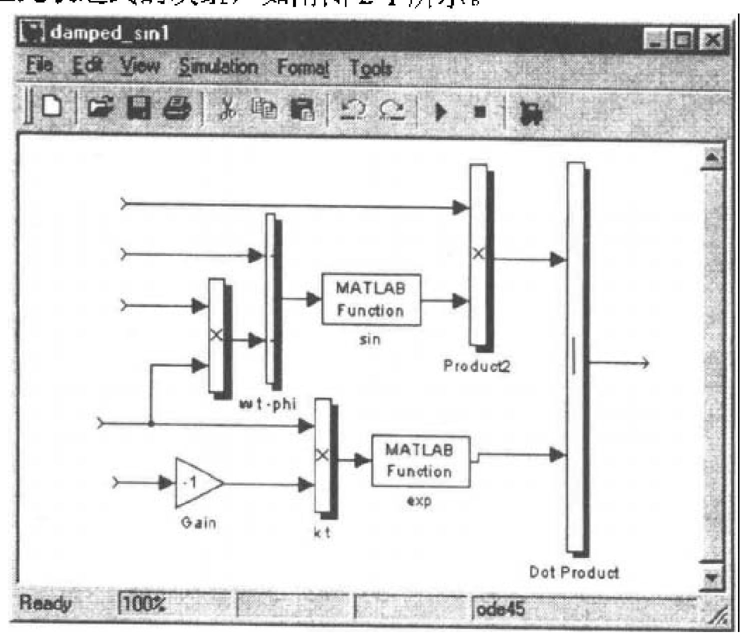
为建造 2 个信号发生器——衰减正弦和伪随机二进制序列（PRBS），我们必须学习块屏蔽。

B.1 衰减正弦信号发生器

我们要制作一个能产生如下表达式的衰减信号的 SIMULINK 块：

$$x(k) = ae^{-kt}\sin(\omega t - \varphi)$$

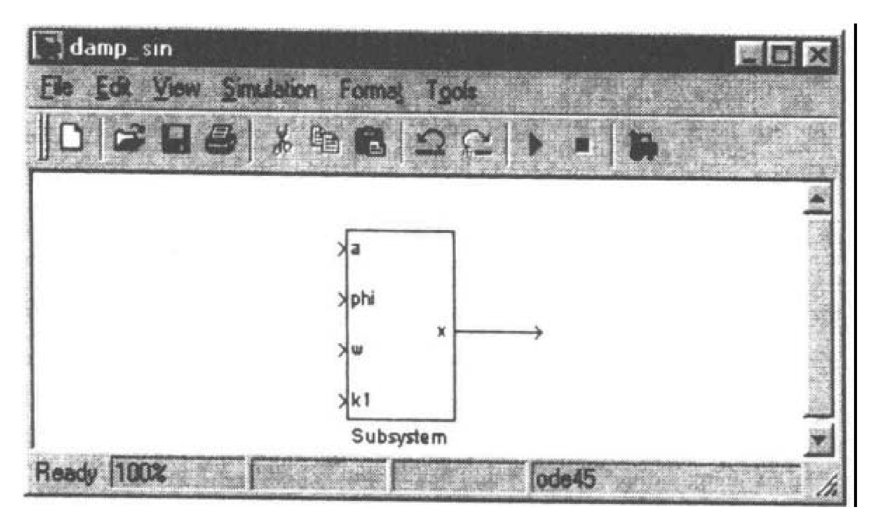
开始构建一个产生此表达式的块组，如附图 2-1 所示。



附图 2-1 产生衰减信号的 SIMULINK 块表达式的块组

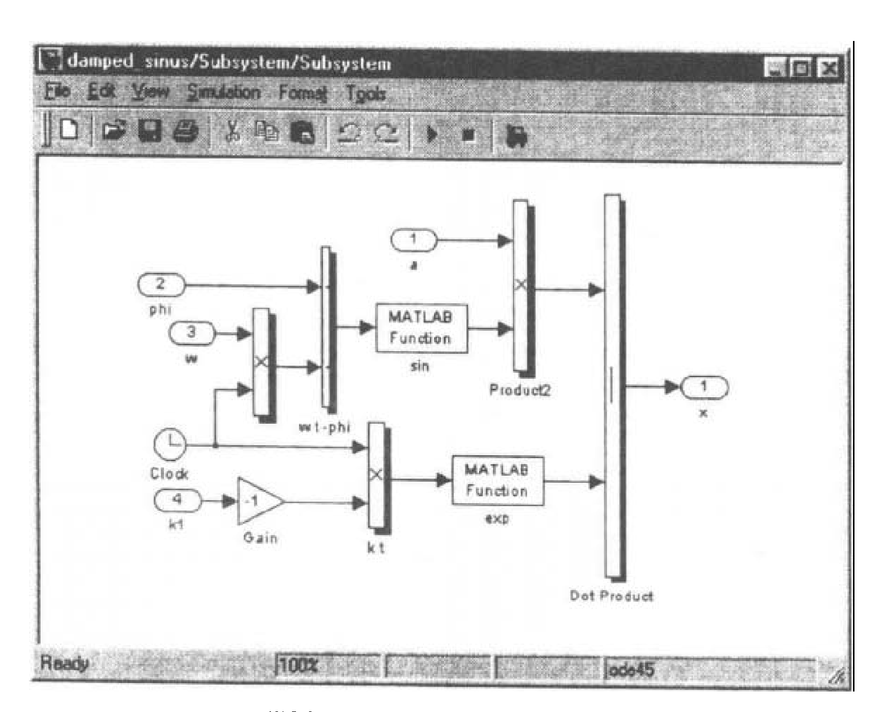
运算符 sinus 和 exponential 由 Nonlinear 库的 MATLAB Function 块来定义。

鼠标选定图示内容后，Edit（编辑）菜单下的 Create Subsystem 选项能产生相应的子系统。双击得到的块，得到系统细节，加上输入输出端口。有必要用不同的名字命名这些端口，如本例中的 a，phi，w，k 以及 x。



附图 2-2　子系统图

子系统可用来产生信号：

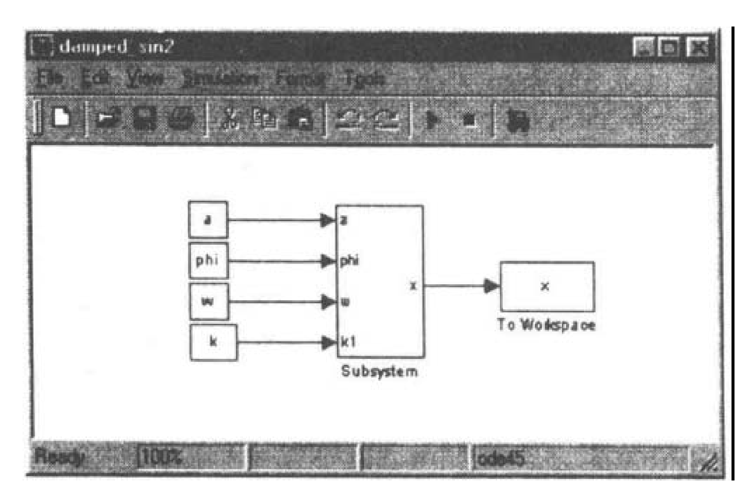


附图 2-3　子系统产生的信号图

$$x(t)=4\mathrm{e}^{-0.2t}\sin 2t$$

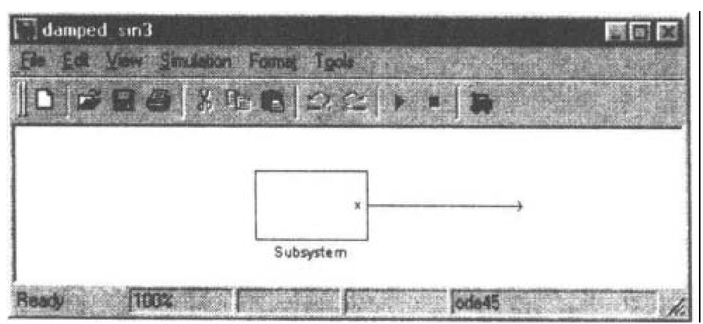
通过全局变量 x 将其传递至工作区。

本例中，表示信号特性的变量是子系统块的常值输入。编辑框能产生用户键入的输入变量的不同值的对话框，编辑帮助文件或绘画曲线。在以前的子系统中，通过不同的名称来代替各种不变的输入值，在同样的常值块里产生那些信号。



附图 2-4　damped_sin2

在选择子系统、删除 To Workspace 块后，通过菜单 Edit 的 Create Subsystem 产生另一个子系统。这个子系统的输入较少，有一个名为 Out1 的输出。双击这个子系统改变它的卷标 x。



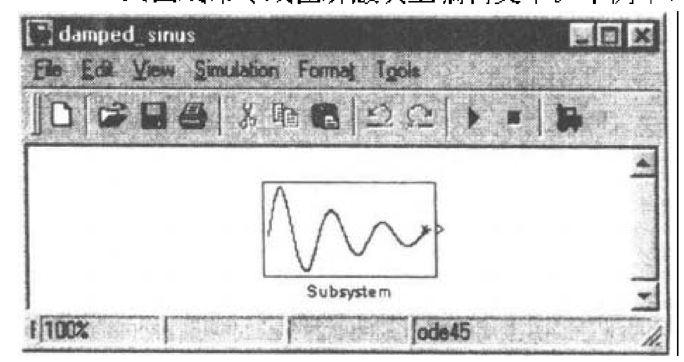
附图 2-5 子系统

在此块中，已用的变量及函数类型不出现。可以通过在它上面画一条正弦曲线和制作对话框来指定使用变量的值，使编辑框个性化。选择图标后，选择菜单 Edit 的 Mask Subsystem 选项，出现 Mask Editor（屏蔽编辑器）窗口，有 3 个菜单：Icon（图标）、Initialization（初始化）、Documentation（文档）。在这个窗口中，我们定义屏蔽参数。

◆　图标菜单

● Mask Type　屏蔽名称或对话框名称，如 Damped sinusoidal signal generator(衰减正弦信号发生器)；

● Drawing commands　画曲线命令或在屏蔽块上编辑文本。本例中，屏蔽图标变为曲线。



附图 2-6　子系统

对于封装图标属性，有 4 种选项：

- Icon frame 使图标框可见或不可见；
- Icon transparency 选项 Transparency 可显示在子系统中定义的输入或输出变量名；
- Icon rotation 根据上面的文本或所画曲线来定义图标旋转属性；
- Drawing co-ordinates 定义图标框坐标为图形窗口。

-Normalized 坐标值固定，左下角为(0,0),右上角为(1,1)；

-Autoscale 自动随所画曲线的最大和最小值协调坐标；

-Pixel 用单位像素画线。

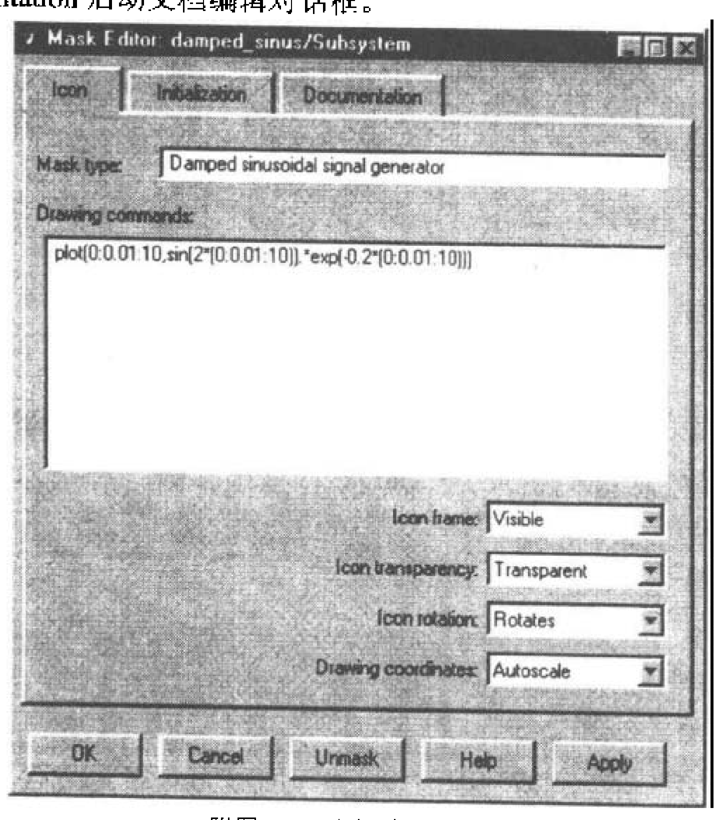
本例中画的是时间自 0 到 10、有选项 Autoscale 的阶跃为 0.01 的曲线。

◆ 菜单 Documentation

这个菜单中定义将进入对话框中的变量。

- Mask type 在对话框标题上出现的屏蔽；
- Block description 出现在对话框中的描述文本块；
- Block help 帮助文档。

菜单 Documentation 启动文档编辑对话框。



附图 2-7 文档编辑对话框

当单击对话框中的按钮 help 时，就会出现块文档 help。要使之有效，需单击 apply 按钮。

◆ 初始化菜单

该菜单定义变量和它们的词义（prompt）。在对话框中键入数字型、链型或布尔型值，对话框中返回它们的词义。变量的数字值可直接键入编辑区或在弹出菜单中进行选择；布尔型使对应复选框为标识（逻辑 1）或保持空（逻辑 0）；变量也可能有链型值，MATLAB

用命令 eval 来解释它。

变量类型由可选择的按钮 Control type 来控制：

- Edit　数值情况；
- Checkbox　复选框；
- Popup　弹出式菜单。

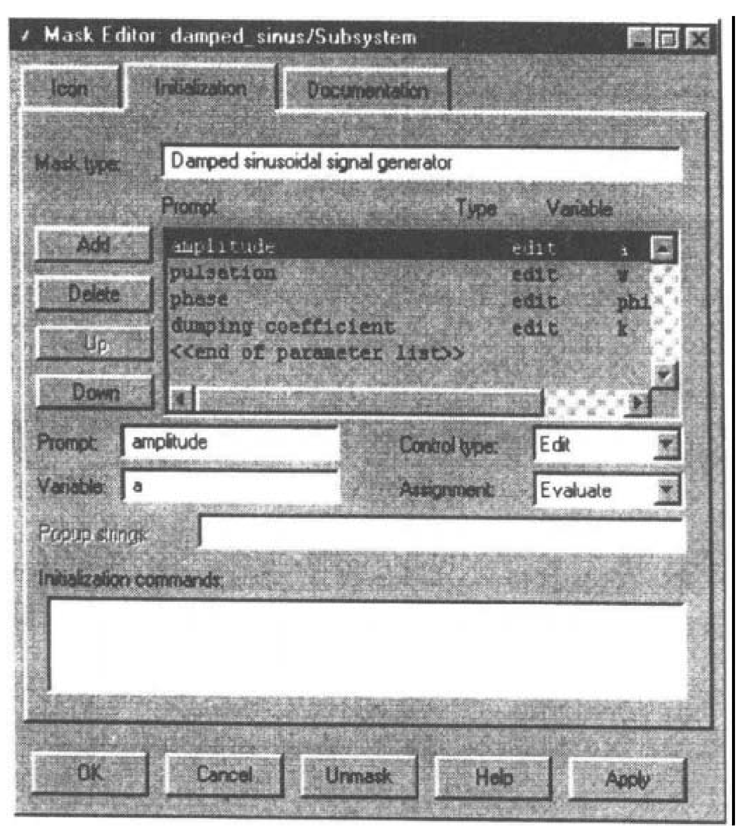
Prompt 对应于变量的词义。在 Variable 区域定义对话框中取特定值的变量名。当填充 Prompt 和 Variable 区域时，此变量的词义、名称和类型在分隔区 end of parameter list 中公布。显示顺序如它在对话框中出现的一样。

按钮 Down 和 Up 用来修改和组织顺序。如果要删除一个变量，必须单击按钮 Delete。

按钮 Assignment 有 2 个选择：MATLAB 直接赋值时的 Evaluate 和变量是字符型变量时的 Literal。

键入的每个变量（prompt 和 name）在选择按钮 Add 后起作用。

附图 2-8 所示为当需要使用编辑的屏蔽块时，这些键入的不同类型变量的选项在对话框中的显示。



附图 2-8　不同类型变量的选项

Prompt 和变量名定义在单击按钮 Add 后进行。如果想在一个弹出菜单中加入一组变量，必须在 Popup zone 区键入它们，并用符号“|”分隔。

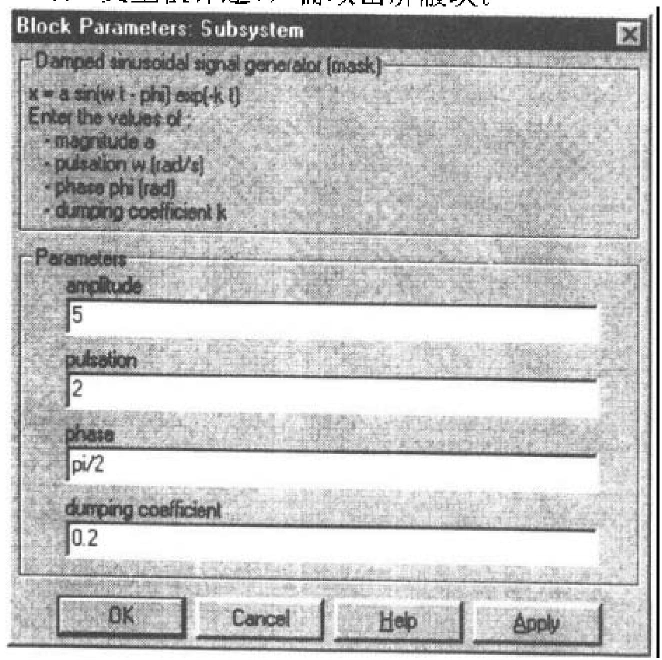
弹出菜单的应用举例在下节中给出，将产生一个伪随机二进制序列（PRBS）。弹出菜单提供对应长度的序列数目的选择。

当产生了屏蔽并想修改 3 个菜单属性中的 1 个时，必须执行菜单 Edit 的选项 Edit Mask。选项 Look Under Mask 能显示整个屏蔽块。

如果想产生如下信号：

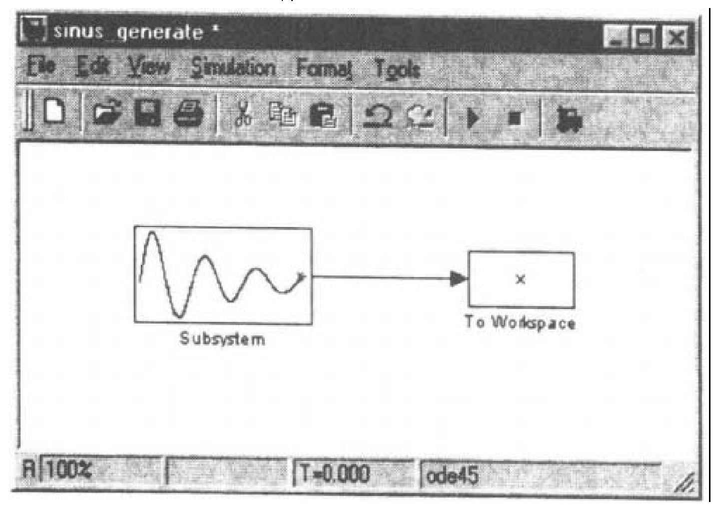
$$x(t) = 5e^{-0.2t}\sin(\omega t - \frac{\pi}{2})$$

为询问不同的对话框区域（变量值详述），需双击屏蔽块。



附图 2-9　块参数

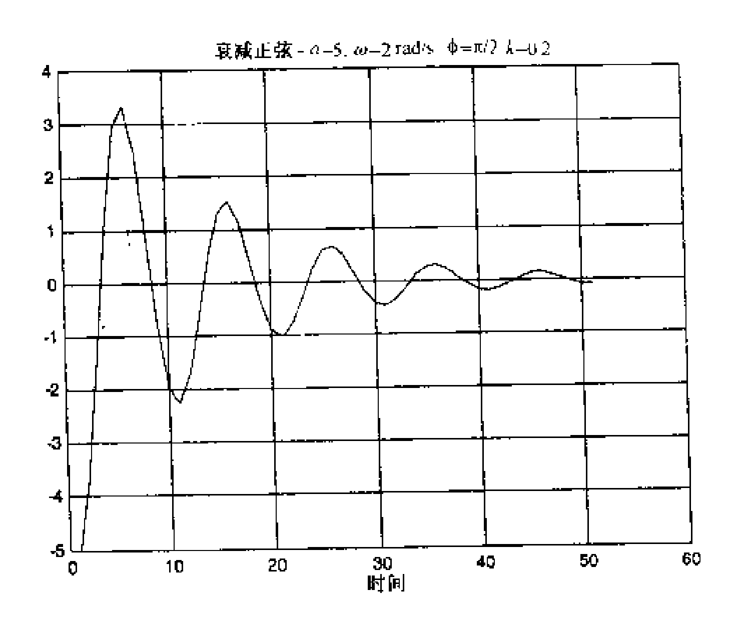
屏蔽产生的信号通过全局变量 x 传到工作区。



附图 2-10　屏蔽产生的信号传至工作区

Curve drawing

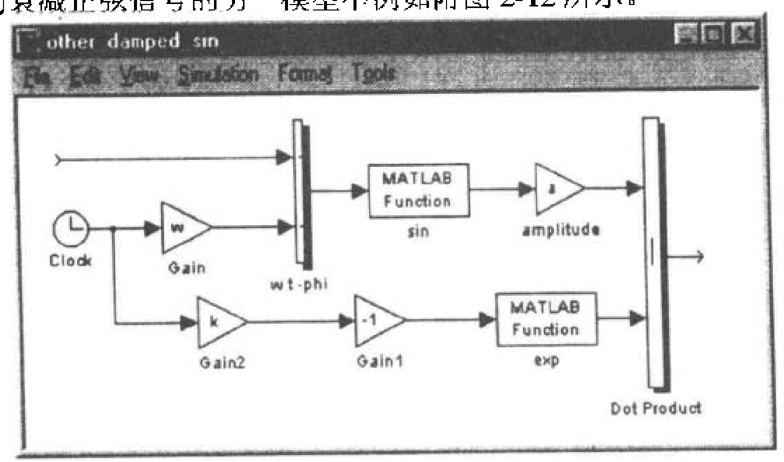
```
>>plot(x)
>>grid
>>title('damped sinus-a=5;w=2rad/s;\phi=\pi/2;k=0.2')
>>xlabel('time')
```



附图 2-11　衰减正弦信号

信号变量的表达由“gain”代替“constant”来表示。

产生同样的衰减正弦信号的另一模型举例如附图 2-12 所示。



附图 2-12　产生衰减正弦信号另一模型举例

该例比前述例子简洁，只有 1 个输入—— 相位 φ 。其他参数，如角频率 ω、衰减系数 k 和幅度 a 在块 gain 中定义。如前所述，由此模型得到的相应子系统同样可使用相同的对话框。

B.2　伪随机二进制序列发生器（PRBS）

伪随机二进制序列在过程控制和信号处理中很有用。

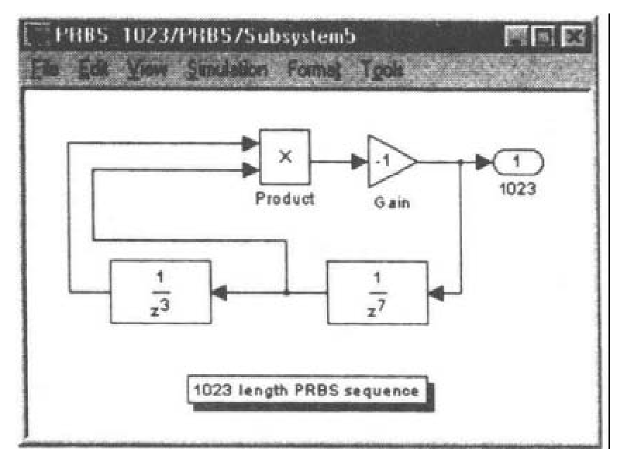
每个 PRBS 序列通过特征多项式定义，1023 长度序列的第 j 个元素如下：

$$u(j)=u(j-7)*u(j-10)$$

为初始化序列的第 1 个值为±1（1023 序列的前 10 位），我们采用传递函数块，定义初始状态和它们的传递延迟。

这些 Discrete Transfer（具有初始状态）在库 Blocksets&Toolboxes/Simulink Extras/Additional

Discrete 中提供。每个序列传列递给子系统。1023 长度的 PRBS 序列的实现如附图 2-13 所示。



附图 2-13　1023 长度 PRBS 序列的实现

为选择这 6 个序列中的 1 个，选用 S 函数 sf_select，将采样时间 T_e 和在最后对话框的弹出菜单中确定的代表序列数目的 k 赋给它。此 S 函数有对应 6 个 PRBS 序列的 6 个输入和 1 个第 k 个序列的输出。

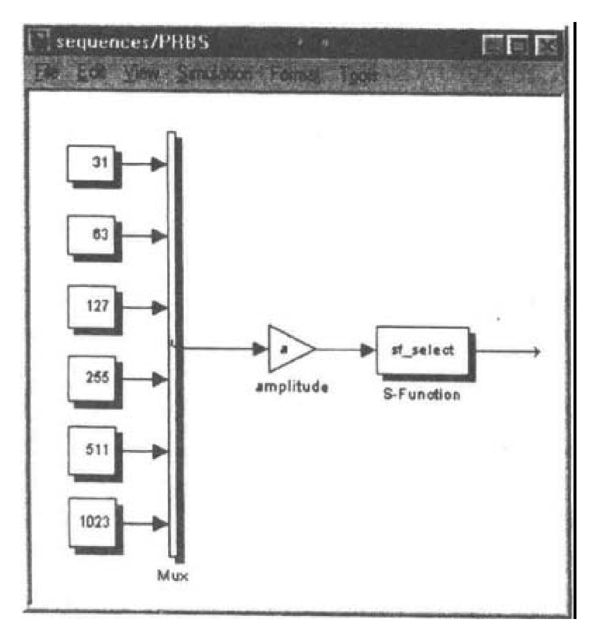
sf_select.m

```
% S-function : PRBS sequence selection
function [sys,x0,str,ts] = sf_select(t,x,u,flag,Te,k)

switch flag
case 0
   k = 1;
   [sys,x0,str,ts] = Initialisation(Te);
case 3
   sys = u(k);
case {1,2,4,9}
   sys = [];
otherwise
   error(['Unhandled flag = ',num2str(flag)]);
end

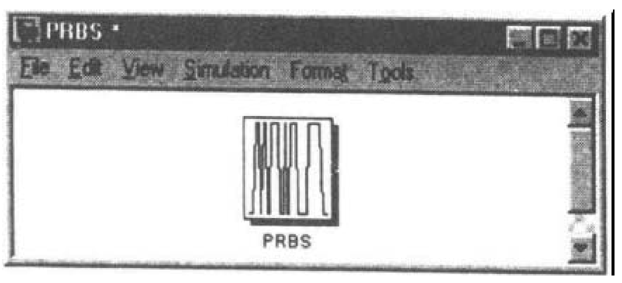
   function [sys,x0,str,ts] = Initialisation(Te)
   sizes = simsizes;
   sizes.NumContStates = 0;
   sizes.NumDiscStates = 0;
   sizes.NumOutputs = 1;% 1 output
   sizes.NumInputs = 6;    % 6 inputs
   sizes.DirFeedthrough = 0;
   sizes.NumSampleTimes = 1;
   sys = simsizes(sizes);
   x0 = [];
   str = [];
   ts = [Te 0];
```

6 个序列块的输出进行多路传输，然后在传给序列选择 S 函数前乘以代表幅度的增益。



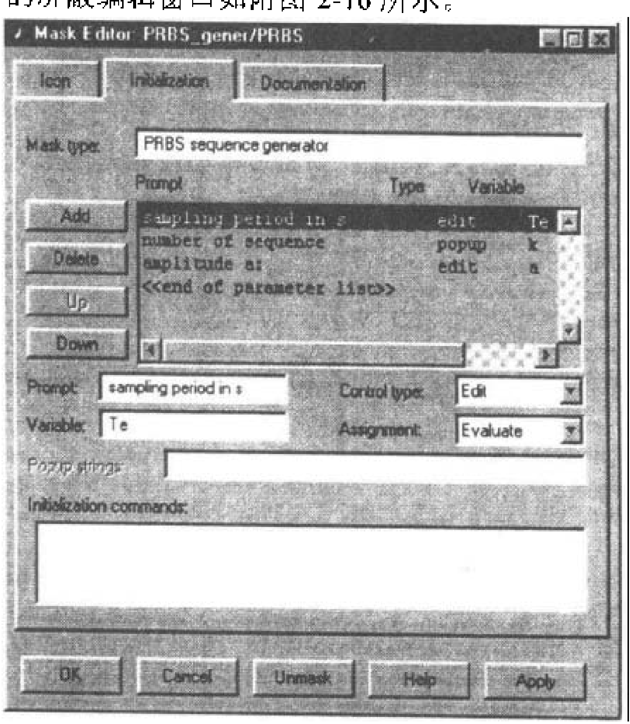
附图 2-14　序列/PRBS

选择传给要被屏蔽的子系统的块组。



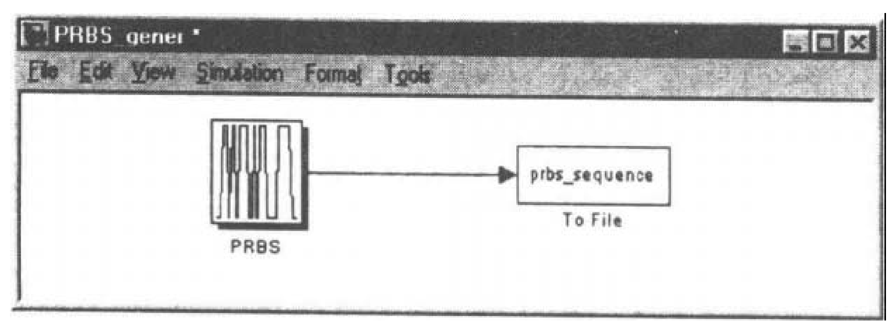
附图 2-15　PRBS

菜单 Initislization 的屏蔽编辑窗口如附图 2-16 所示。



附图 2-16　屏蔽编辑器

得到的屏蔽了系统用于产生长为 1023 的第 6 个序列。



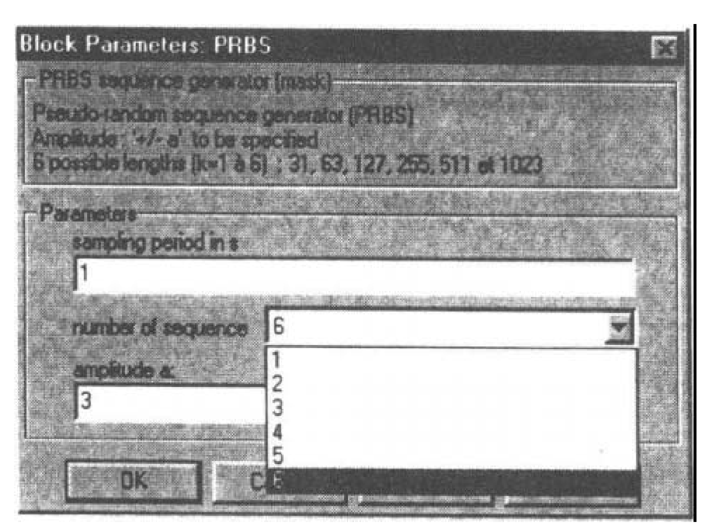
附图 2-17　PRBS 发生器

产生的信号送往数据文件 prbs_sequence.mat。

双击 PRBS 屏蔽块，打开对话框，在其中确定：

- 以秒为单位的采样周期 T_s；
- 弹出菜单中的 6 个序列数目；
- 幅度为 3。

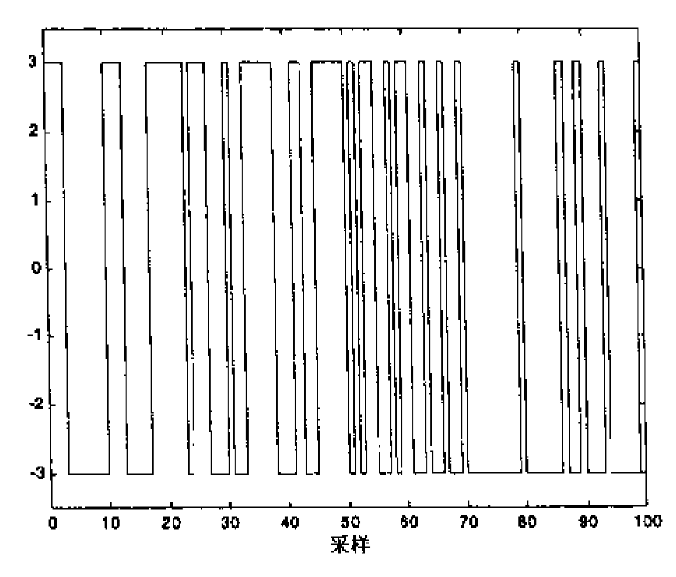
采样周期值传给低层块，所以顺序选择 S 函数和 Discrete Transfer Fcn（具有初始状态）块用于产生不同长度的 PRBS 模型。



附图 2-18　块参数

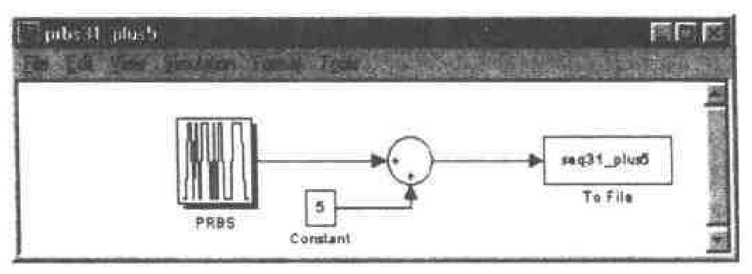
在这个窗口中，认可用于数目顺序选择的弹出菜单。下面的命令行可画出产生的 PRBS 序列。

```
>>load prbs_sequence, t=prbs(1,: ); prbs=prbs(2,: );
>>stirs(t,prbs(,  axis([0  100  -3.5  3.5]])
>>title('PRBS sequence of length 1023 and amplitude 3')
>>xlabel('samles')
```



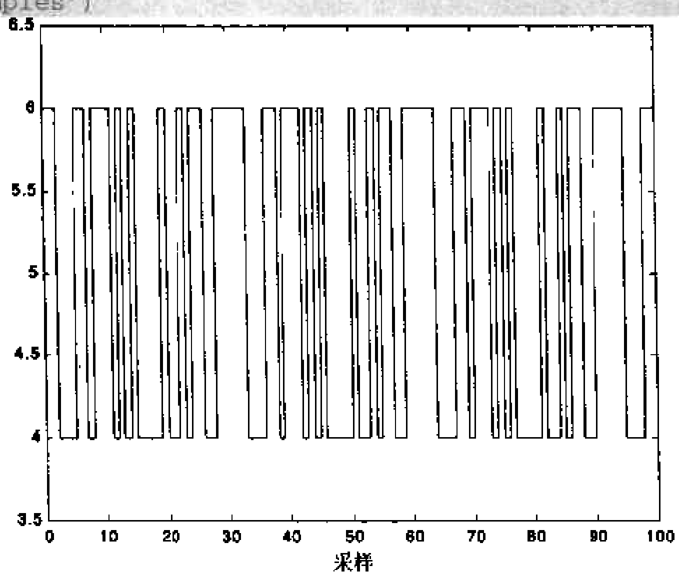
附图 2-19　长为 1023、幅度为 3 的 PRBS 序列

下图中，产生长为 31（顺序数 1）、在常量 5 上下的 PRBS 序列。



附图 2-20　prbs31_plus5 发生器

```
>>stairs(t,prbs)
>>axis([0  100  3  7])
>>axis([0  100  3.5  6.5])
>>title('PRBS sequence of length 31,amplitude 1,around 5 constant value')
>>xlabel('samples')
```



附图 2-21　长为 31、幅度为 1、在常数 5 上下的 PRBS 序列